VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Minimalizace odchozího stupně vrcholu v téměř pravidelném grafu

Minimalize of outdegree of vertices into an almost regular graph

2017 Bc. Jakub Závada

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra aplikované matematiky

Zadání diplomové práce

Student: **Bc. Jakub Závada**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 1103T031 Výpočetní matematika

Téma: Minimalizace odchozího stupně vrcholu v téměř pravidelném grafu
Minimalize of outdegree of vertices into an almost regular graph

Jazyk vypracování: čeština

Zásady pro vypracování:

Při složitých aplikovaných numerických úlohách je nutné výpočet paralelizovat a rovnoměrně zatížit jednotlivé výpočetní uzly. Optimální rozmístění jednotlivých podúkolů na uzly řeší grafová úloha, kdy chceme orientovat jednoduchý graf tak, aby největší odchozí stupeň výsledného orientovaného grafu byl co nejmenší.

Pokud jde o úlohy ve 2D, tak graf, který situaci modeluje, je rovinný. V případě rovinných grafů je nám už známo, že graf lze orientovat v lineárním čase tak, že odchozí stupeň každého vrcholu nepřevýší 3. Tedy, na každém uzlu je zpracován výpočet pro nejvýše tři úseky dvojice oblastí.

Otázka je, jak rychle optimalizovat rozmístění zátěže v případě 3D úloh. Tzn. hledáme nejnižší odchozí stupeň při orientaci obecných grafů. Vyřešit tuto otázku obecně je těžké, proto se zatím zaměříme na tzv. téměř pravidelné grafy. Téměř pravidelné grafy můžeme definovat například tak, že pro každý vrchol grafu x platí d-1 <= deg(x) <= d+1$, pro pevně zvolené d.

Naznačme postup. V prvé řadě musíme zjistit, jak souvisí tzv. arboricita grafu s pravidelností grafu v závislosti na jeho struktuře (velmi hrubý odhad jsme schopni stanovit ze známého vzorce pro arboricitu obecného grafu, my tento odhad ovšem chceme zpřesnit, a tudíž musíme brát v potaz i strukturu grafu). Z toho pak odvodíme alespoň odhad pro nejnižší odchozí stupeň grafu.

Seznam doporučené odborné literatury:

R. Haas - Characterizations of Arboricity of Graphs

M. Chrobak, D. Eppstein - Planar Orientations with Low Out-Degree and Compaction of Adjacency Matrices

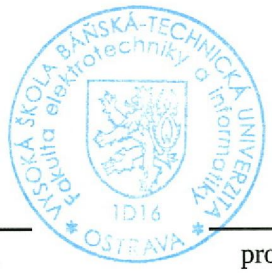
Dále, dle pokynů vedoucího práce

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **RNDr. Michael Kubesa, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017



doc. RNDr. Jiří Bouchala, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. dubna 2017 .....

Rád bych na tomto místě poděkoval RNDr. Michaelu Kubesovi, Ph.D. za odborné vedení, rady, připomínky a veškerou další pomoc při tvorbě této diplomové práce.

Abstrakt

Tato práce se zabývá minimalizací nejvyššího odchozího stupně vrcholu v téměř pravidelném grafu. Téměř pravidelný graf je graf, v němž se nejnižší a nejvyšší stupeň vrcholu liší maximálně o 2. Nejprve ukážeme, jak určit optimální hodnotu nejvyššího odchozího stupně v daném téměř pravidelném grafu, a pak i způsob, jak orientaci hran s minimálním nejvyšším odchozím stupněm nalézt. Součástí práce je i algoritmus napsaný v Matlabu, který pro zadaný téměř pravidelný graf určí optimální orientaci jeho hran. Práce by měla mít využití při paralelních numerických výpočtech, kde rozdělujeme zadaný objekt do několika podoblastí a na každém vlákně je zpracovávána jedna podoblast spolu s několika částmi své hranice. Orientace hran duálního grafu (každá podoblast je v duálním grafu reprezentována jedním vrcholem a hrana je mezi dvěma vrcholy, pokud dané podoblasti mají společnou hranici) určí, v rámci kterého vlákna se bude výpočet na hranici mezi dvěma oblastmi provádět tak, aby na žádném vlákně nebylo příliš mnoho výpočtů.

Klíčová slova: téměř pravidelný graf, orientace hran, odchozí stupeň, uzavřený eulerovský tah

Abstract

This thesis is focused on the minimization of the maximum outdegree of vertex in an almost regular graph. The almost regular graph is a graph in which the difference between maximum and minimum degree of vertex is at most two. First we will show how to determine the optimal value of maximum outdegree of given almost regular graph and then we will show the way, how we can find the orientation with minimized maximum outdegree. We also give Matlab algorithm which finds optimal edge orientation for given almost regular graph. Our results can be used in parallel numerical computing where given object is divided into subdomains and each thread is used for computing on one subdomain and on some parts of the boundary of this subdomain. The edge orientation of a dual graph (each subdomain is represented by one vertex and there is an edge between two vertices if the subdomains represented by these vertices are neighboring) determines which thread will be used for computing on the boundary between any two vertices as we want to minimize the largest number of tasks on a thread.

Key Words: almost regular graph, edge orientation, outdegree, Eulerian circuit

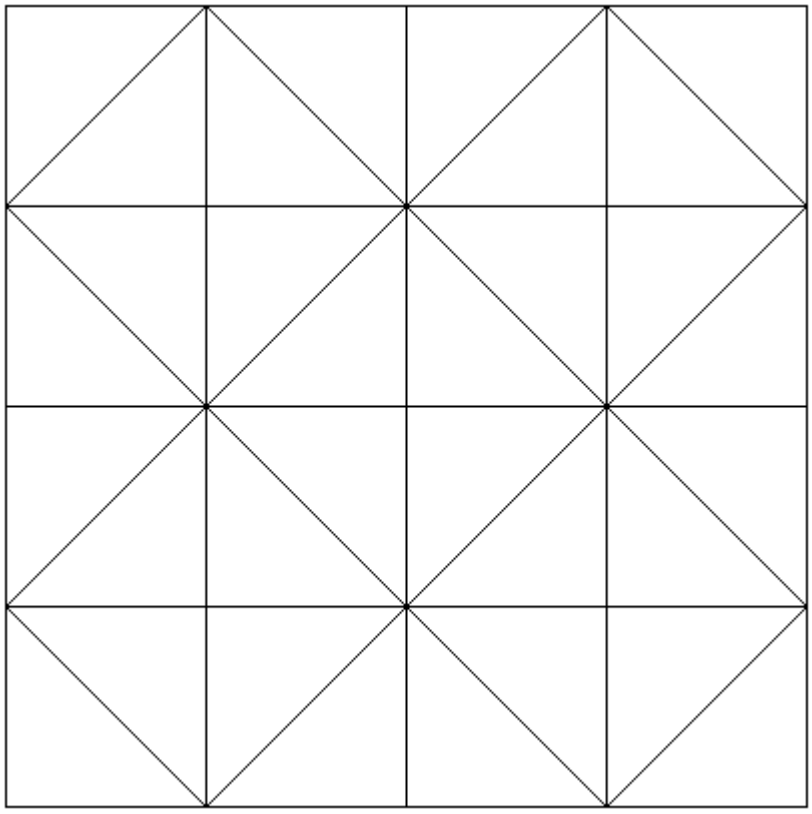
Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Při aplikovaných numerických výpočtech se zkoumaný objekt rozdělí na konečný počet oblastí (viz Obrázek 1). Konkrétní výpočty se potom provádějí na těchto oblastech a na vnitřních hranicích mezi jednotlivými oblastmi (předpokládáme, že na okraji objektu jsou zadány okrajové podmínky). Jde často o složité výpočty a je třeba tyto výpočty paralelizovat, abychom ušetřili výpočetní čas. Chceme, aby na jednom výpočetním uzlu byla zpracovávána jedna oblast s nějakými částmi své hranice tak, aby bylo rozmístění podúloh na jednotlivé výpočetní uzly co nejrovnoměrnější, tzn. aby na žádném výpočetním uzlu nebyl spolu s oblastí zpracováván zbytečně velký počet částí jejích hranic. Optimální rozmístění podúloh na jednotlivé výpočetní uzly řeší následující grafová úloha.

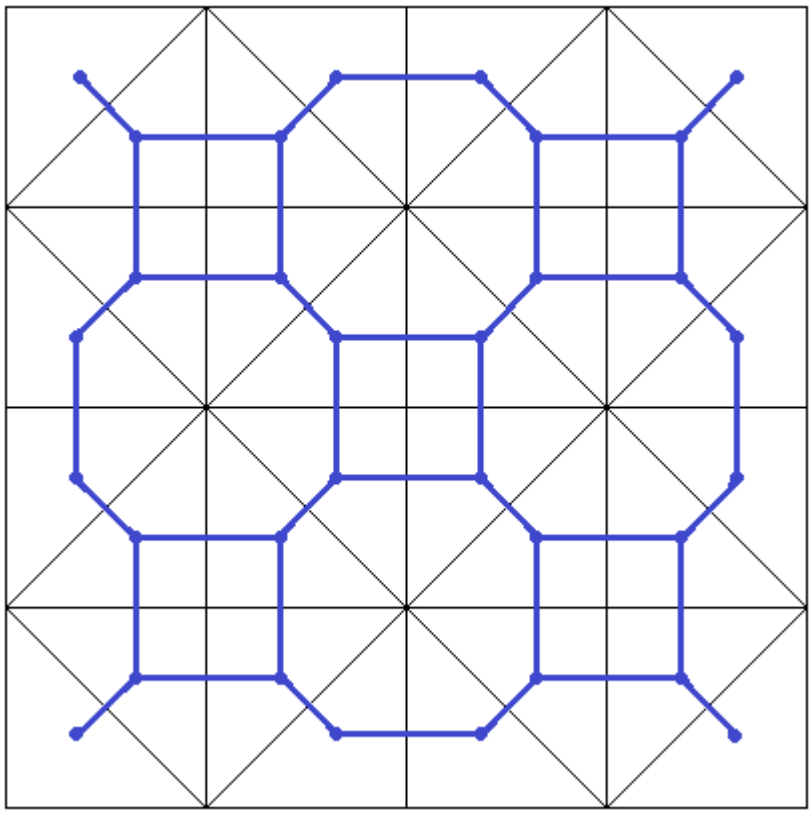


Obrázek 1: Rozdělení čtverce na trojúhelníkové oblasti

Poznámka 1 V této části budeme používat některé grafové pojmy intuitivně, jejich přesné definice uvedeme v příští kapitole.

Mějme nějaké rozdělení zkoumaného objektu na oblasti (viz Obrázek 1). K takovému rozdělení sestavíme tzv. *duální graf* G tak, že vrcholy grafu G jsou jednotlivé oblasti a dva vrcholy

jsou spojeny hranou právě tehdy, když reprezentují oblasti, které mají společnou hranici (viz Obrázek 2). Hrany dáváme jen mezi oblasti, jejichž společná hranice má nenulovou délku, případně u 3D objektů nenulový obsah. Například když rozdělíme krychli na menší krychličky, zajímají nás ty dvojice krychliček, které mají společnou stranu, ale už ne ty, které mají pouze společnou hranu nebo bod. Každou hranu grafu G nějak zorientujeme, čímž vytvoříme graf orientovaný a řekneme, že s oblastí reprezentovanou vrcholem x budou zpracovávány na stejném výpočetním uzlu právě ty hranice, které jsou v orientovaném grafu reprezentovány *odchozími hranami* - to jsou ty, které jsou orientovány z vrcholu x do jiného vrcholu. Počet všech odchozích hran incidentních s vrcholem x je *odchozí stupeň vrcholu* x.



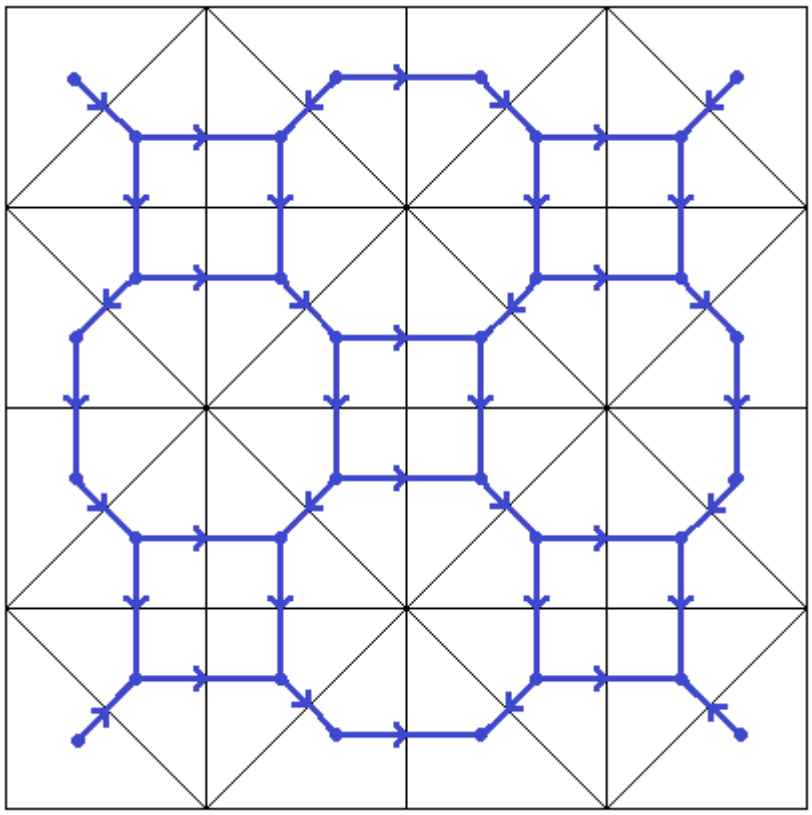
Obrázek 2: Duální graf

Je známo, že součet odchozích stupňů všech vrcholů v orientovaném grafu je roven počtu jeho hran. Počet hran orientovaného grafu značíme $|A|$ a počet vrcholů $|V|$. *Průměrný odchozí stupeň* vrcholu v grafu je tedy dán číslem $\frac{|A|}{|V|}$ a alespoň jeden vrchol musí mít odchozí stupeň větší nebo roven tomuto číslu. Proto nejvyšší odchozí stupeň vrcholu musí být alespoň $\frac{|A|}{|V|}$.

Chceme-li tedy v daném rozdělení zkoumaného objektu paralelizovat výpočty na oblastech a jejich hranicích do jednotlivých výpočetních uzlů co nejrovnoměrněji, pak to znamená, že duální graf se snažíme orientovat tak, aby nejvyšší odchozí stupeň vrcholu v grafu byl co nejmenší, tudíž co nejbližší průměrnému odchozímu stupni $\frac{|A|}{|V|}$.

Úkol nalézt optimální orientaci hran není jednoduchý, proto se řešení problému často zaměřuje pouze na určitý typ grafů. V této práci se zaměříme na *téměř pravidelné grafy*. Jsou to grafy, kde rozdíl mezi největším a nejmenším stupněm je nejvýše 2. Všimněme si, že duální graf na Obrázku 2 je téměř pravidelný. Duální grafy pro různá rozdělení různých objektů budou často téměř pravidelné, proto má smysl se téměř pravidelnými grafy zabývat. Duální graf na Obrázku 2 je navíc i *rovinný* (nekříží se v něm hrany). Z dřívější práce [1] už víme, že hrany jakéhokoli rovinného grafu je možné orientovat tak, aby nejvyšší odchozí stupeň byl nejvýše 3. Hrany grafu na Obrázku 2 je ale možné orientovat i tak, aby nejvyšší odchozí stupeň byl 2 (viz Obrázek 3). V této práci se naučíme orientovat hrany libovolného téměř pravidelného grafu tak, aby nejvyšší odchozí stupeň byl optimální (tedy nejnižší možný).

V Kapitole 3 uvedeme a dokážeme věty, které nám řeknou, jaký je optimální nejvyšší odchozí stupeň v daném téměř pravidelném grafu. V Kapitole 4 se naučíme v *eulerovských grafech* hledat *uzavřený eulerovský tah* (tah, kterým projdeme všechny hrany a vrcholy grafu, každá hrana se přitom musí použít právě jednou). V Kapitole 5 pak získané znalosti využijeme k hledání optimální orientace hran v libovolném téměř pravidelném grafu. Nakonec v Kapitole 6 otestujeme naprogramovaný algoritmus na různých téměř pravidelných grafech a porovnáme jej s už dříve známým algoritmem.



Obrázek 3: Orientace hran duálního grafu

2 Základní pojmy

Definice a věty v této kapitole jsou z velké části inspirovány definicemi a větami z [2].

Definice 1 Jednoduchý graf G *je uspořádaná dvojice* (V, E)*, kde* V *je neprázdná množina* vrcholů *a* E *je množina* hran*. Množinu vrcholů grafu* G *označujeme* $V(G)$ *a množinu hran grafu* G *označujeme* $E(G)$*. Množina hran je nějaká množina dvouprvkových podmnožin množiny* V*.*

Grafu $G(V, E, w)$, kde V a E mají stejný význam jako v definici jednoduchého grafu a w je vektor, který každé hraně grafu G přiřazuje nějaké číslo (ohodnocení), říkáme *jednoduchý hranově ohodnocený graf.*

Počtu vrcholů v grafu říkáme *řád grafu*, počet hran je *velikost grafu.* Vrcholy v grafu G budeme označovat písmeny, čísly nebo $v_1, v_2, \ldots, v_n$, kde n je řád grafu G. Místo hrany $\{x, y\}$ píšeme zkráceně xy. O hraně xy říkáme, že je s vrcholy x a y *incidentní.* Vrcholy $x, y \in V(G)$ jsou *sousední*, jestliže $xy \in E(G)$, a *nezávislé*, pokud $xy \notin E(G)$.

Existují i další typy grafů, pro potřeby tohoto textu musíme definovat ještě graf orientovaný.

Definice 2 Orientovaný graf *nebo také* digraf D *je uspořádaná dvojice* (V, A)*, kde* V *je neprázdná množina vrcholů a* A *je nějaká podmnožina kartézského součinu* $V \times V$*. Množinu* A *nazýváme množinou* orientovaných hran*. Množinu vrcholů orientovaného grafu* D *označujeme* $V(D)$ *a množinu orientovaných hran orientovaného grafu* D *označujeme* $A(D)$*. Vrchol* x *nazýváme* počátečním vrcholem *a vrchol* y koncovým vrcholem *hrany* (x, y)*. Dále říkáme, že hrana* (x, y) *je* odchozí hranou *z vrcholu* x *a* příchozí hranou *do vrcholu* y*.* Smyčka *je orientovaná hrana, která má stejný počáteční i koncový vrchol, tedy hrana typu* (x, x)*.*

Definice orientovaného grafu umožňuje, aby v jednom orientovaném grafu byla jak hrana (x, y), tak i hrana (y, x). V této práci se ale s takovými grafy setkávat nebudeme, neboť budeme vždy vycházet z jednoduchého (neorientovaného) grafu a v něm pro každou hranu zvolíme orientaci. V našich orientovaných grafech tedy nebudou ani smyčky, jelikož v jednoduchých grafech smyčky nejsou.

Grafy znázorňujeme obvykle pomocí diagramů. Vrchol si můžeme představit jako bod v rovině, přičemž na jednom místě může být vždy nejvýše jeden vrchol, a hranu si můžeme představit jako křivku mezi dvěma vrcholy. V případě orientované hrany je orientace hrany znázorněna šipkou směrem od příchozího vrcholu ke koncovému. V počítači se grafy znázorňují nejčastěji pomocí *matice sousednosti*, často se používá také *matice incidence.*

Definice 3 *O grafu* G *řekneme, že je* planární (rovinný)*, jestliže existuje takové jeho zakreslení do roviny, že s výjimkou vrcholů nemají libovolné dvě hrany žádné další společné body.*

Planární graf je tedy graf, v němž se na žádném místě nekříží hrany.

Poznámka 2 Někteří autoři rozlišují pojmy rovinný a planární graf. Za rovinný graf nepovažují graf, který lze zakreslit bez křížení hran do roviny, ale graf, který máme zakreslený bez křížení hran do roviny. V takovém případě se vlastně jedná spíše o vlastnost daného zakreslení než o vlastnost grafu.

Definice 4 *Nechť je dán jednoduchý graf G řádu n. Označme vrcholy grafu čísly $1, 2, ..., n$.* Maticí sousednosti *grafu G rozumíme takovou čtvercovou matici $A(G)$ n-tého řádu, která má na pozici (i, j), kde $i, j \in \{1, 2, \cdots, n\}$, jedničku právě tehdy, když je v grafu G hrana ij. V případě, že v grafu G hrana ij není, je na pozici (i, j) v matici $A(G)$ nula.*

Matice sousednosti jednoduchého grafu je symetrická a na diagonále má samé nuly. Matici sousednosti orientovaného grafu definujeme tak, že jednička na pozici (i, j) bude pouze v případě, že v grafu je orientovaná hrana (i, j), tedy hrana s počátečním vrcholem i a koncovým vrcholem j. Matice sousednosti orientovaného grafu tudíž nemusí být symetrická a diagonála nemusí být tvořena pouze nulami, neboť jestliže je v grafu smyčka (i, i), na pozici (i, i) v matici sousednosti bude jednička.

Definice 5 *Mějme grafy G a H. Graf H je* podgrafem *grafu G právě tehdy, když $V(H) \subseteq V(G)$ a $E(H) \subseteq E(G)$. Graf G je zároveň* nadgrafem *grafu H. Pokud z grafu G odebereme pouze nějaké vrcholy a kromě hran incidentních s odebranými vrcholy neodebereme žádné další, graf H je* indukovaný podgraf *grafu G. Pokud platí $V(G) = V(H)$, pak je podgraf H* faktorem *grafu G.*

Podgraf grafu G tedy vznikne odebráním libovolného (i nulového) počtu vrcholů a hran z grafu G. Pokud odebíráme nějaký vrchol, odebereme rovněž všechny hrany incidentní s tímto vrcholem.

Definice 6 Stupněm vrcholu *x v jednoduchém grafu rozumíme počet hran incidentních s vrcholem x. Stupeň vrcholu x značíme* $\deg(x)$. *Největší stupeň v jednoduchém grafu G se značí $\Delta(G)$, nejmenší stupeň $\delta(G)$, přičemž*

$$\Delta(G) = \max_{v \in V(G)} \{\deg(v)\} \quad \text{a} \quad \delta(G) = \min_{v \in V(G)} \{\deg(v)\}.$$

Chceme-li zdůraznit, že se jedná o stupeň vrcholu x v konkrétním grafu G, stupeň vrcholu x můžeme značit $\deg_G(x)$.

Věta 1 *V každém jednoduchém grafu $G(V, E)$ platí*

$$\sum_{v \in V} \deg(v) = 2\,|E|\,.$$

Důkaz Tvrzení dokážeme indukcí vzhledem k počtu hran.

Indukční základ: Pro grafy bez hran platí

$$\sum_{v \in V} \deg(v) = 2\,|E| = 0.$$

Indukční krok: Předpokládejme, že tvrzení platí pro všechny jednoduché grafy s nejvýše k hranami. Mějme libovolný jednoduchý graf $G = (V, E)$, kde $|E| = k + 1$. Z grafu G odebereme jednu hranu $xy \in E$. Vznikne graf $G'(V', E') = G(V, E) - xy$, kde $|V'| = |V|$ a $|E'| = |E| - 1 = k$. Můžeme předpokládat, že platí

$$\sum_{v \in V'} \deg(v) = 2\,|E'| = 2k.$$

Vrátíme-li do grafu G' hranu xy, dostaneme jednoduchý graf G a platí

$$\deg_G(x) = \deg_{G'}(x) + 1 \quad \text{a} \quad \deg_G(y) = \deg_{G'}(y) + 1.$$

Odtud

$$\sum_{v \in V} \deg(v) = \sum_{v \in V'} \deg(v) + 2.$$

Proto

$$\sum_{v \in V} \deg(v) = 2k + 2 = 2(k + 1) = 2\,|E|\,.$$

Věta platí i v grafu s $k + 1$ hranami a důkaz je tak podle principu matematické indukce hotový.

■

Uvedené větě se často říká *princip sudosti*. Důležitým důsledkem této věty je, že v každém grafu musí být sudý počet vrcholů lichého stupně. Součet stupňů všech vrcholů je roven $2\,|E|$, je tedy sudý. Kdyby počet vrcholů lichého stupně byl lichý, byl by pak lichý i součet stupňů všech vrcholů, neboť součet lichého počtu lichých čísel je vždy lichý a přičtením libovolného počtu sudých čísel k lichému číslu dostaneme opět liché číslo. To je ale ve sporu s Větou 1, počet vrcholů lichého stupně v libovolném grafu G je tedy vždy sudý.

Definice 7 *Mají-li všechny vrcholy v grafu G stejný stupeň d, nazýváme tento graf* d-pravidelným grafem *nebo pouze* pravidelným grafem. *O grafu G, pro který platí $\Delta(G) - \delta(G) \leq 2$, budeme hovořit jako o* téměř pravidelném grafu.

V orientovaných grafech se s pojmem stupeň vrcholu (nebo také *celkový stupeň vrcholu*) můžeme setkat také. Navíc je zde definován ještě *příchozí stupeň vrcholu* jako počet příchozích hran do vrcholu x, značíme $\deg^-(x)$, a *odchozí stupeň vrcholu* jako počet odchozích hran z vrcholu x, značíme $\deg^+(x)$. Nejmenší odchozí stupeň vrcholu v orientovaném grafu D zna-

číme $\delta^+(D)$, největší odchozí stupeň $\Delta^+(D)$, nejmenší příchozí stupeň $\delta^-(D)$ a největší příchozí stupeň $\Delta^-(D)$.

V této práci budeme pojem největší odchozí stupeň používat i u jednoduchých grafů. $\Delta^+(G)$ bude mít v případě jednoduchého grafu G následující význam.

Definice 8 *Nechť D_G je množina všech orientovaných grafů D, které mohou vzniknout orientováním všech hran jednoduchého grafu G. Pak nejvyšší odchozí stupeň $\Delta^+(G)$ jednoduchého grafu G definujeme jako*

$$\Delta^+(G) = \min_{D \in D_G} \{\Delta^+(D)\}.$$

Věta 2 *Pro každý orientovaný graf $D(V, A)$ platí*

$$\sum_{v \in V} \deg^+(v) = \sum_{v \in V} \deg^-(v) = |A|.$$

Důkaz Je poměrně zřejmé, že tato věta platí. Ukážeme si důkaz indukcí vzhledem k počtu hran.

Indukční základ: Pro grafy bez hran platí

$$\sum_{v \in V} \deg^+(v) = \sum_{v \in V} \deg^-(v) = |A| = 0.$$

Indukční krok: Předpokládejme, že tvrzení platí pro všechny orientované grafy s nejvýše k hranami. Mějme libovolný orientovaný graf $D = (V, A)$, kde $|A| = k + 1$. Z grafu D odebereme jednu hranu $(x, y) \in A$. Vznikne graf $D'(V', A') = D(V, A) - (x, y)$, kde $|V'| = |V|$ a $|A'| = |A| - 1 = k$. Můžeme předpokládat, že platí:

$$\sum_{v \in V'} \deg^+(v) = \sum_{v \in V'} \deg^-(v) = |A'| = k.$$

Vrátíme-li do grafu D' hranu (x, y), dostaneme orientovaný graf D a platí

$$\deg^+_D(x) = \deg^+_{D'}(x) + 1 \quad \text{a} \quad \deg^+_D(y) = \deg^+_{D'}(y) + 1.$$

Odtud

$$\sum_{v \in V} \deg^+(v) = \sum_{v \in V'} \deg^+(v) + 1 \quad \text{a} \quad \sum_{v \in V} \deg^-(v) = \sum_{v \in V'} \deg^-(v) + 1.$$

Proto

$$\sum_{v \in V} \deg^+(v) = \sum_{v \in V} \deg^-(v) = k + 1 = |A|.$$

Vidíme tedy, že uvedená věta platí i v orientovaném grafu D s $k + 1$ hranami.

■

Definice 9 *Nerostoucím způsobem seřazenou posloupnost stupňů všech vrcholů grafu G nazýváme* stupňovou posloupností *grafu G.* Grafová posloupnost *je posloupnost taková, že existuje graf, pro který platí, že tato posloupnost je jeho stupňovou posloupností.*

Definice 10 *Posloupnost vrcholů a hran $(v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n)$ v jednoduchém grafu G je* sled, *jestliže pro všechna $i = 1, 2, \ldots, n$ jsou vrcholy v_{i-1} a v_i incidentní s hranou e_i.* Tah *je sled, ve kterém se každá hrana grafu G vyskytuje nejvýše jednou. Tah je* uzavřený, *jestliže $v_0 = v_n$.* Cesta *je sled, ve kterém se každý vrchol grafu G vyskytuje nejvýše jednou.* Cyklus *je sled, ve kterém se vyskytuje nejvýše jednou každý vrchol grafu G s výjimkou vrcholu $v_0 = v_n$, který se zde vyskytuje dvakrát.*

Vzhledem k tomu, že v jednoduchém grafu je mezi danými dvěma vrcholy vždy maximálně jedna hrana (grafy, které mohou mít mezi dvěma vrcholy více než jednu hranu, nazýváme *multigrafy*), zápis sledu, tahu nebo cesty v jednoduchém grafu lze zjednodušit tak, že ze zápisu vynecháme hrany. Hrana sledu, tahu nebo cesty je v jednoduchém grafu vždy jednoznačně určena koncovými vrcholy a je postačující, když zapíšeme pouze posloupnost po sobě jdoucích vrcholů. Namísto posloupnosti $(v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n)$ proto budeme používat kratší zápis $(v_1, v_2, \ldots, v_n)$.

Definice 11 Uzavřený eulerovský tah *je uzavřený tah v grafu G, který obsahuje všechny vrcholy i hrany grafu G.* Eulerovský graf *je graf, v němž lze nalézt uzavřený eulerovský tah. Jestliže v grafu G existuje tah, který obsahuje všechny jeho vrcholy i hrany, ale $v_0 \neq v_n$, nazýváme tento tah* otevřeným eulerovským tahem.

Při hledání nejlepšího orientování hran (ve smyslu minimalizace největšího odchozího stupně) budeme často využívat vlastnosti tzv. eulerovských grafů.

Definice 12 Eulerovský graf *je graf, v němž lze nalézt uzavřený eulerovský tah.*

Definice 13 *O grafu $G(V, E)$ říkáme, že je* souvislý, *jestliže z libovolného vrcholu $v \in V(G)$ existuje cesta do všech ostatních vrcholů grafu G. Graf, který není souvislý, je* nesouvislý.

Je-li graf G souvislý a graf $G - xy$ nesouvislý, o hraně xy řekneme, že tato hrana je most.

Definice 14 *$L(V', E')$ je* komponenta *grafu $G(V, E)$ právě tehdy, když L je souvislý podgraf grafu G a v grafu G neexistuje sled z vrcholu $v \in V'(L)$ do žádného vrcholu u, pro který platí $u \in V(G)$ a zároveň $u \notin V'(L)$.*

Věta 3 *Jednoduchý graf G je eulerovský právě tehdy, když je souvislý a všechny jeho vrcholy mají sudý stupeň.*

Důkaz Dokazujeme ekvivalenci, postupně dokážeme obě implikace.

- „ $\Longrightarrow$ “: Začneme implikací „ $\Longrightarrow$ “, jejíž důkaz je jednodušší. Dokazujeme tedy následující větu. Jestliže je graf G eulerovský, pak je souvislý a všechny jeho vrcholy mají sudý stupeň.

  Eulerovský graf je graf, v němž existuje uzavřený eulerovský tah. Předpokládejme, že graf G je nesouvislý. Zahájíme tah v nějaké komponentě. V průběhu tohoto tahu se žádným způsobem nemůžeme dostat do dalších komponent grafu G. Není proto možné projít tímto tahem všechny vrcholy a hrany grafu G. Sporem jsme tak dokázali, že eulerovský graf musí vždy být souvislý.

  Dále dokažme, že všechny vrcholy grafu G mají sudý stupeň. Zahajme tah v nějakém vrcholu x. K návratu do vrcholu x potřebujeme dvě hrany (první hranou vrchol x opustíme, druhou se do něj vrátíme). Pokud stále zbývají hrany incidentní s vrcholem x, opět vrchol x opustíme pomocí nějaké ze zbývajících hran. Do vrcholu x se takto vracíme vždy po použití sudého počtu hran s ním incidentních a po použití lichého počtu takových hran jsme vždy v nějakém sousedním vrcholu vrcholu x. Vrchol x tedy musí mít sudý stupeň, aby v grafu mohl existovat uzavřený eulerovský tah. Z podobného důvodu musí být sudého stupně i všechny další vrchol y grafu G. Vždy po použití lichého počtu hran incidentních s vrcholem y budeme ve vrcholu y (pokud y není počáteční vrchol tahu). Tah v tu chvíli nemůže být uzavřený, protože začal v jiném vrcholu. Vrchol y tedy musí být sudého stupně, aby vždy po navštívení vrcholu y bylo možné další hranou tento vrchol opustit. Všechny vrcholy eulerovského grafu proto musí mít sudý stupeň.

- „ $\Longleftarrow$ “: Důkaz opačné implikace je složitější. Nyní chceme dokázat následující větu. Jestliže je graf G souvislý a všechny jeho vrcholy mají sudý stupeň, pak je graf G eulerovský.

  Pokusme se v souvislém grafu G, který má pouze vrcholy sudého stupně, vytvořit nějaký uzavřený tah. Tah zahájíme v libovolném vrcholu x. Pokud je x jediný vrchol grafu G, stupeň tohoto vrcholu je 0 (nulu považujeme za sudé číslo) a v grafu G je triviální uzavřený eulerovský tah složený pouze z vrcholu x. Je-li v grafu G více vrcholů, zvolíme libovolnou hranu xy incidentní s vrcholem x. Takové hrany jsou v grafu minimálně dvě. Pomocí hrany xy se přesuneme do vrcholu y. Z vrcholu y se nějakou hranou yz přesuneme do vrcholu z. Už použitou hranu xy nemůžeme použít znovu, platí proto $x \neq z$. Vybíráme postupně další hrany, dokud se nevrátíme zpět do počátečního vrcholu x. Graf má konečný počet hran a není možné v průběhu tahu „uvíznout“ v nějakém jiném vrcholu, neboť stupeň všech vrcholů je sudý a vždy po navštívení nějakého vrcholu (kromě počátečního vrcholu x) zbývá lichý počet hran incidentních s tímto vrcholem, které jsme ještě neprošli. Vždy tedy zbývá nějaká hrana, pomocí které tento vrchol můžeme opustit. Proto se po konečném počtu kroků musíme vrátit zpět do vrcholu x.

  Do vrcholu x se vrátíme buď po projití všech hran, což by znamenalo, že jsme nalezli uzavřený eulerovský tah, nebo po návratu do vrcholu x ještě nějaké hrany zbývají a nalezli jsme pouze tah, který je uzavřený, ale není eulerovský. Pokud nějaké hrany zbývají, pak

pro graf $G' = G - E(t)$, kde $E(t)$ je množina všech hran právě nalezeného uzavřeného tahu, stále platí, že všechny vrcholy grafu G' jsou sudého stupně. Tento graf G' už nemusí být souvislý, ale v každé komponentě K grafu G' je alespoň jeden vrchol w, který je součástí nalezeného uzavřeného tahu. Pokud by v grafu G' byla taková komponenta L, že žádný vrchol komponenty L není součástí už nalezeného tahu, znamenalo by to, že graf G nebyl souvislý (odstraňujeme pouze hrany, které jsou součástí nalezeného tahu, jeho součástí tedy musí být i koncové vrcholy těchto hran, proto pokud odstraněním nějaké hrany porušíme souvislost, graf se rozdělí na dvě komponenty a jeden koncový vrchol odstraněné hrany je v jedné z těchto dvou komponent, zatímco druhý je ve druhé komponentě, v každé komponentě tak zůstává vrchol, který je součástí nalezeného tahu). V tomto vrcholu w můžeme zahájit další uzavřený tah, kterým projdeme nějaké hrany komponenty K a opět se vrátíme do vrcholu w. Takto nalezený tah t' můžeme připojit k tahu t v místě, kde se v tahu t poprvé vyskytuje vrchol w. Tím tah t rozšíříme o další hrany, přičemž stále platí, že t je uzavřený tah. Opět je možné, že komponenta K se po odebrání hran tahu t' rozdělila na několik dalších komponent. Znovu ale platí, že v každé nové komponentě jsou pouze vrcholy sudého stupně, můžeme tedy stejným způsobem nalézt v každé komponentě další uzavřený tah, který připojíme k už nalezenému tahu t. Takto můžeme pokračovat, dokud v tahu t nebudou všechny hrany grafu G.

■

Vidíme, že uvedený důkaz je konstruktivní - ukazuje, jak lze uzavřený eulerovský tah najít. Použitý důkaz je inspirován Hierholzerovým algoritmem, kterým se budeme zabývat v Podkapitole 4.2.

3 Výpočet $\Delta^+(G)$ v téměř pravidelném grafu G

Problém minimalizace nejvyššího odchozího stupně vrcholu v libovolném jednoduchém hranově ohodnoceném grafu $G = (V, E, w)$ je NP-úplný problém [3]. V libovolném jednoduchém grafu $G = (V, E)$ s neohodnocenými hranami, případně obecněji v grafu $G = (V, E, w)$, kde všechny hrany mají stejné ohodnocení, je uvedený problém řešitelný v polynomiálním čase [3]. Pro některé třídy grafů, například pro stromy (souvislé grafy, které neobsahují žádný cyklus), existují i rychlejší algoritmy. Konkrétně pro stromy existuje jednoduchý algoritmus s lineární časovou složitostí, který nalezne orientaci hran s nejvyšším odchozím stupněm 1 v libovolném stromu [3]. Rovinnými grafy se velmi podrobně zabývá práce [1], v níž je ukázán způsob, jak v libovolném grafu najít orientaci hran tak, aby nejvyšší odchozí stupeň nebyl vyšší než 3. My se nyní zaměříme na téměř pravidelné grafy.

Mějme libovolný téměř pravidelný graf $G(V, E)$. Naším úkolem je orientováním všech hran grafu G vytvořit orientovaný graf D tak, aby odchozí stupeň $\Delta^+(D)$ byl co nejmenší. Začneme d-pravidelnými grafy. Poté se budeme zabývat grafy, které mají pouze vrcholy stupňů d a $d-1$, a nakonec grafy, pro něž platí $\Delta(G)-\delta(G) = 2$. V této kapitole se budeme zabývat pouze otázkou, jaká je hodnota $\Delta^+(G)$ v daném téměř pravidelném grafu. Nalezením vhodné orientace hran se budeme zabývat později. Vzhledem k rozdílu mezi vlastností $\Delta^+(G)$ jednoduchého grafu G a vlastností $\Delta^+(D)$ orientovaného grafu D připomínáme Definici 8, v níž definujeme $\Delta^+(G)$.

Jako dolní odhad pro $\Delta^+(G)$ budeme vždy používat průměrný odchozí stupeň, přesněji řečeno horní celou část průměrného odchozího stupně v orientovaném grafu D, který vznikne orientací všech hran grafu G (je zřejmé, že průměrný odchozí stupeň bude stejný pro libovolnou orientaci hran grafu G, na konkrétní podobě grafu D tedy v tuto chvíli nezáleží). Průměrný odchozí stupeň v grafu D budeme označovat $\varnothing \deg^+(D)$.

Pro orientovaný graf $D(V, A)$ řádu n platí

$$\varnothing \deg^+(D) = \frac{1}{n} \sum_{v \in V} \deg^+(v) = \frac{1}{n} |A| = \frac{|A|}{n}.$$

Není možné, aby platilo

$$\forall v \in V : \deg^+(v) < \varnothing \deg^+(D),$$

neboť dostaneme

$$\sum_{v \in V} \deg^+(v) < |A|,$$

což je vyloučeno (viz Věta 2), a proto musí platit

$$\exists v \in V : \deg^+(v) \geq \varnothing \deg^+(D).$$

Odtud

$$\Delta^+(G) \geq \varnothing \deg^+(D) = \frac{|A|}{n}.$$

Z předešlého také plyne, že

$$\Delta^+(G) \geq \left\lceil \varnothing \deg^+(D) \right\rceil.$$

3.1 Pravidelný graf

Mějme d-pravidelný graf. Nejprve určíme dolní odhad pro $\Delta^+(G)$ jako $\lceil \varnothing \deg^+(D) \rceil$, kde $D(V, A)$ je orientovaný graf vzniklý libovolnou orientací všech hran grafu $G(V, E)$. V grafu G je n vrcholů a všechny vrcholy jsou stupně d.

Platí $|A| = \frac{dn}{2}$ a po dosazení za $|A|$ dostáváme dolní odhad

$$\left\lceil \varnothing \deg^+(D) \right\rceil = \left\lceil \frac{|A|}{n} \right\rceil = \left\lceil \frac{dn}{2n} \right\rceil = \left\lceil \frac{d}{2} \right\rceil.$$

Věta 4 *V libovolném d-pravidelném grafu G platí*

$$\Delta^+(G) = \left\lceil \frac{d}{2} \right\rceil.$$

Důkaz Dolní odhad hodnoty $\Delta^+(G)$ už známe. Ukážeme, že dolního odhadu lze vhodným orientováním hran v d-pravidelném grafu vždy dosáhnout. Postup bude záviset na paritě d. Jednodušší bude případ, kdy d je sudé. Začneme proto tímto případem.

- Sudé d - Všechny vrcholy mají sudý stupeň d, v souvislém grafu G tedy můžeme najít uzavřený eulerovský tah. Pokud graf G není souvislý, budeme každou komponentu řešit jako samostatný graf a najdeme uzavřený eulerovský tah v každé komponentě. Otázkou, jak uzavřený eulerovský tah najít, se budeme zabývat později. Teď si vystačíme s informací, že takovýto tah v grafu G existuje. Orientací hran ve směru tohoto tahu docílíme, že každý vrchol bude mít $\frac{d}{2}$ odchozích hran a $\frac{d}{2}$ příchozích hran, jelikož do každého vrcholu během tohoto tahu vždy nejprve jednou hranou vstoupíme, a hned poté jej další hranou zase opustíme. Výjimkou je začátek a konec tahu, kdy nejprve počáteční vrchol opustíme a nakonec se do něj po vyčerpání všech hran vrátíme a už v něm zůstaneme. Existuje tedy orientace, při které mají všechny vrcholy odchozí stupeň roven $\frac{d}{2}$. Vzhledem k tomu, že dolní odhad nejvyššího odchozího stupně (tedy průměrný odchozí stupeň), je $\left\lceil \frac{d}{2} \right\rceil = \frac{d}{2}$, můžeme s jistotou říci, že pro libovolný d-pravidelný graf G, kde d je sudé, platí

$$\Delta^+(G) = \frac{d}{2} = \left\lceil \frac{d}{2} \right\rceil.$$

- Liché d - V tomto případě budeme graf modifikovat. Pokusíme se přidat do grafu G vrcholy a hrany tak, aby nově vzniklý graf G' byl eulerovský. Není-li graf G souvislý, budeme každou komponentu považovat za samostatný graf a úlohu si tak rozdělíme do několika

podúloh. Víme, že všechny vrcholy eulerovského grafu mají sudý stupeň. Dále víme, že každý graf má sudý počet vrcholů lichého stupně, což je důsledek Věty 1. To nám umožňuje sestrojit graf G' tak, že libovolným způsobem spárujeme vrcholy lichého stupně. Spárování provedeme následujícím způsobem. Pro každé dva vrcholy x a y, které chceme spárovat, přidáme do grafu jeden nový vrchol z. Dále do grafu přidáme hrany xz a yz. V grafu G' nyní nejsou žádné vrcholy lichého stupně, a proto je možné v grafu G' nalézt uzavřený eulerovský tah. Hrany můžeme orientovat ve směru tohoto uzavřeného eulerovského tahu. Tím vznikne orientace, pro kterou platí

$$\forall v \in V(G') : \deg^+(v) = \deg^-(v).$$

Tento orientovaný graf označme D'. Z orientovaného grafu D' nyní získáme orientovaný graf D tak, že odebereme vrcholy, které jsme do grafu G přidali při tvorbě grafu G'. Graf D je tedy vlastně indukovaný podgraf grafu D'. Odebráním vrcholů a hran není možné nejvyšší odchozí stupeň zvýšit, určitě tedy platí $\Delta^+(D) \leq \Delta^+(D')$.

V grafu G je $n = 2k$ vrcholů stupně d, v grafu G' bude $2k$ vrcholů stupně $d+1$ a k vrcholů stupně 2. Orientováním hran ve směru nalezeného uzavřeného eulerovského tahu dostaneme graf D', pro který platí $\Delta^+(D') = \frac{d+1}{2} = \lceil \frac{d}{2} \rceil$. Toto číslo se shoduje s dolním odhadem $\Delta^+(G)$. Proto platí

$$\Delta^+(G) = \Delta^+(D') = \frac{d+1}{2} = \left\lceil \frac{d}{2} \right\rceil.$$

Dostáváme tedy požadovaný výsledek bez ohledu na to, jestli je d sudé nebo liché, a dokázali jsme tak Větu 4.

■

3.2 Graf, který má pouze vrcholy stupňů d a $d-1$

Obdobným postupem jako v předchozí podkapitole dojdeme ke stejnému výsledku i v případě, že se nejnižší a nejvyšší stupeň grafu G liší o jedničku. Výše dokázaná rovnost tedy platí nejen pro d-pravidelné grafy, ale i pro grafy, v nichž jsou kromě vrcholů stupně d ještě navíc vrcholy stupně $d-1$.

Věta 5 *Nechť d je libovolné přirozené číslo. Pak pro libovolný téměř pravidelný graf G, který má pouze vrcholy stupňů d a $d-1$, přičemž v grafu je alespoň jeden vrchol stupně d, platí*

$$\Delta^+(G) = \left\lceil \frac{d}{2} \right\rceil.$$

Důkaz Nejprve určíme dolní odhad pro $\Delta^+(G)$. Už víme, že pro d-pravidelný graf G a graf D vzniklý libovolnou orientací hran grafu G platí

$$\Delta^+(G) \geq \left\lceil \varnothing \deg^+(D) \right\rceil = \left\lceil \frac{d}{2} \right\rceil .$$

V $(d-1)$-pravidelném grafu analogicky platí pro libovolnou orientaci hran

$$\Delta^+(G) \geq \left\lceil \varnothing \deg^+(D) \right\rceil = \left\lceil \frac{d-1}{2} \right\rceil .$$

My máme v grafu G vrcholy stupně d i vrcholy stupně $d-1$. Průměrný stupeň vrcholu musí být v takovém případě vyšší než průměrný stupeň vrcholu v $(d-1)$-pravidelném grafu a zároveň nižší než průměrný stupeň vrcholu v d-pravidelném grafu. Průměrný stupeň vrcholu bude tedy nějaké číslo z otevřeného intervalu $(d-1, d)$. Pokud připustíme i nulový počet vrcholů stupně $d-1$ (Věta 5 to narozdíl od nulového vrcholu stupně d umožňuje), uvedený interval bude zprava uzavřený.

Libovolnou orientací všech hran grafu G vznikne orientovaný graf D. Průměrný odchozí stupeň vrcholu $\varnothing \deg^+(D)$ je poloviční oproti průměrnému celkovému stupni, neboť každá hrana přispívá jedničkou do odchozího stupně jednoho vrcholu a do celkového stupně dvou vrcholů. Platí tedy

$$\varnothing \deg^+(D) \in \left(\frac{d-1}{2}, \frac{d}{2} \right\rangle .$$

Snadno se můžeme přesvědčit, že pro libovolné číslo x z uvedeného intervalu platí

$$\lceil x \rceil = \left\lceil \frac{d}{2} \right\rceil .$$

Platí tedy

$$\left\lceil \varnothing \deg^+(D) \right\rceil = \left\lceil \frac{d}{2} \right\rceil .$$

Proto platí

$$\Delta^+(G) \geq \left\lceil \frac{d}{2} \right\rceil .$$

Dále z grafu G vytvoříme graf G' tak, že vrcholy lichého stupně spárujeme a přidáme hrany a vrcholy stejně jako v předchozí podkapitole v případě d-pravidelných grafů, kde d je liché.

- Sudé d - V případě sudého d jsou v grafu G' nyní pouze vrcholy stupňů 2 a d, a v grafu G' lze proto nalézt uzavřený eulerovský tah, s jehož pomocí najdeme takovou orientaci hran D', aby platilo

$$\Delta^+(G) = \Delta^+(D') = \frac{d}{2} = \left\lceil \frac{d}{2} \right\rceil .$$

- Liché d - V případě lichého d jsou v grafu G' pouze vrcholy stupňů 2, $d-1$ a $d+1$. V grafu G' lze nalézt uzavřený eulerovský tah. Díky tomuto tahu nalezneme orientaci hran D', pro kterou platí

$$\Delta^+(G) = \Delta^+(D') = \frac{d+1}{2} = \left\lceil \frac{d}{2} \right\rceil .$$

■

3.3 Graf s vlastností $\Delta(G) - \delta(G) = 2$

Mějme graf G splňující následující podmínky. Pro libovolné $d \geq 2$ jsou v grafu G pouze vrcholy stupňů d, $d-1$ a $d-2$, přičemž v grafu G je alespoň jeden vrchol stupně d a alespoň jeden vrchol stupně $d-2$. Vrcholy stupně $d-1$ v grafu být mohou, ale nemusí.

Situaci si opět rozdělíme na dva případy. Začneme tím jednodušším, kdy d je sudé. Stejným postupem dojdeme ke stejnému výsledku jako v předchozích podkapitolách.

Věta 6 *Nechť G je graf, pro který platí $\Delta(G) - \delta(G) = 2$, a nechť $d = \Delta(G)$ je sudé číslo. Pak platí*

$$\Delta^+(G) = \frac{d}{2} = \left\lceil \frac{d}{2} \right\rceil .$$

Důkaz Podobnými úvahami jako v předchozí kapitole můžeme opět dojít k tomu, že dolní odhad pro $\Delta^+(G)$ je $\lceil \frac{d}{2} \rceil = \frac{d}{2}$. V grafu jsou pouze vrcholy stupňů d, $d-1$ a $d-2$, průměrný celkový stupeň tedy bude z intervalu $(d-2, d)$, a průměrný odchozí stupeň proto bude z intervalu $\left(\frac{d-2}{2}, \frac{d}{2}\right)$. V případě sudého d platí pro libovolné x z uvedeného intervalu

$$\lceil x \rceil = \left\lceil \frac{d}{2} \right\rceil .$$

V grafu G', který vytvoříme stejným způsobem jako v předchozích případech, nebudou vrcholy jiných stupňů než 2, $d-2$ a d. V grafu jsou pouze vrcholy sudého stupně, existuje tedy uzavřený eulerovský tah (v nesouvislém grafu považujeme každou komponentu za samostatný graf a uzavřený eulerovský tah existuje v každé komponentě), s jehož pomocí můžeme orientovat hrany tak, že nejvyšší odchozí stupeň v orientovaném grafu D' bude roven $\frac{d}{2}$. Proto platí

$$\Delta^+(G) = \frac{d}{2} = \left\lceil \frac{d}{2} \right\rceil .$$

■

Doposud byl výsledek vždy stejný. To svádí k očekávání, že i v posledním případě dojdeme k obvyklému výsledku $\Delta^+(G) = \left\lceil \frac{d}{2} \right\rceil$. Skutečnost je ale jiná.

Věta 7 *Nechť G je graf, pro který platí $\Delta(G) - \delta(G) = 2$, a nechť $d = \Delta(G)$ je liché číslo. Pak platí*

$$\Delta^+(G) \in \left\{ \left\lfloor \frac{d}{2} \right\rfloor, \left\lceil \frac{d}{2} \right\rceil \right\}.$$

Důkaz Všimněme si, že dolní odhad $\Delta^+(G)$ tentokrát není jednoznačný, neboť závisí na tom, zda je v grafu G více vrcholů stupně d než vrcholů stupně $d-2$. Jestliže je v grafu G stejný počet vrcholů stupně d a $d-2$, pak průměrný stupeň bude přesně $d-1$. Pokud v grafu bude větší počet vrcholů stupně d než stupně $d-2$, pak už průměrný stupeň bude větší než $d-1$ a pro průměrný odchozí stupeň po orientování hran bude platit $\varnothing \deg^+(D) > \frac{d-1}{2}$. Zároveň víme, že průměrný stupeň musí být nižší než d, proto průměrný odchozí stupeň musí být nižší než $\frac{d}{2}$. Platí tedy $\varnothing \deg^+(D) \in \left(\frac{d-1}{2}, \frac{d}{2}\right)$ a pro každé číslo x z uvedeného intervalu platí $\lceil x \rceil = \left\lceil \frac{d}{2} \right\rceil$, dolní odhad pro $\Delta^+(G)$ je proto $\left\lceil \frac{d}{2} \right\rceil$. V tuto chvíli můžeme pokračovat obdobně jako v případě sudého d a stejnými myšlenkami, které je zbytečné zde opakovat, dojdeme k tomu, že dolního odhadu vždy dosáhneme. Platí tedy

$$\Delta^+(G) = \left\lceil \frac{d}{2} \right\rceil.$$

Pokud ale v grafu G není více vrcholů stupně d než vrcholů stupně $d-2$, pak bude platit $\varnothing \deg^+(D) \leq \frac{d-1}{2}$. Zároveň víme, že musí platit $\varnothing \deg^+(D) > \frac{d-2}{2}$, průměrný odchozí stupeň je tedy z intervalu $\left(\frac{d-2}{2}, \frac{d-1}{2}\right\rangle$. Pro každé x z uvedeného intervalu a liché d platí $\lceil x \rceil = \frac{d-1}{2} = \left\lfloor \frac{d}{2} \right\rfloor$. Dolní odhad je tedy o 1 nižší než v předchozím odstavci. Zde selhává způsob, jakým jsme doposud vše dokazovali, neboť úpravou grafu G na G' pomocí přidání vrcholů a hran a následným orientováním hran grafu G' dostaneme jako obvykle orientovaný graf D', pro který platí $\Delta^+(D') = \left\lceil \frac{d}{2} \right\rceil$. Pro liché d ale platí $\left\lceil \frac{d}{2} \right\rceil = \left\lfloor \frac{d}{2} \right\rfloor + 1$. Odstraněním vrcholů přidaných při tvorbě grafu G' sice odstraníme i nějaké hrany, nemusí to ale stačit k tomu, aby pro vzniklý graf D platilo $\Delta^+(D) = \left\lfloor \frac{d}{2} \right\rfloor$.

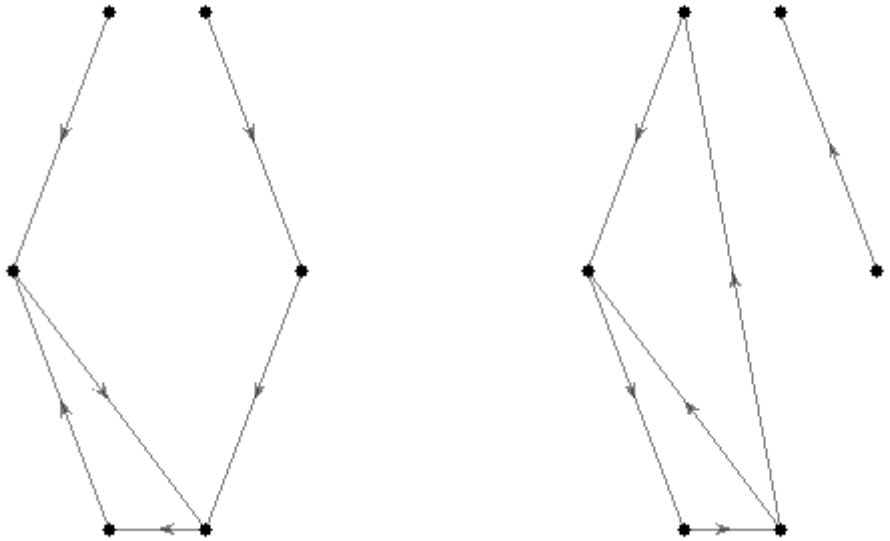
Nabízí se otázka, zda existuje graf s vlastnostmi z Věty 7, pro který platí $\Delta^+(G) = \left\lfloor \frac{d}{2} \right\rfloor$. Na příkladu můžeme ukázat, že ano. Dokonce se výsledný minimalizovaný nejvyšší odchozí stupeň může lišit i u dvou různých grafů se stejnou stupňovou posloupností, viz Obrázek 4, kde graf D_1 vznikl orientací hran grafu G_1 a graf D_2 vznikl orientací hran grafu G_2, přičemž grafy G_1 a G_2 mají stejnou stupňovou posloupnost. Pro grafy na Obrázku 4 platí $\Delta^+(D_1) = 1$ a $\Delta^+(D_2) = 2$.

Graf D_1 splňuje požadavek pro optimální orientaci hran grafu G_1, proto platí

$$\Delta^+(G_1) = \Delta^+(D_1) = 1.$$

Vzhledem k nízkému počtu hran se můžeme vyzkoušením všech možných orientací hran grafu G_2 poměrně snadno přesvědčit, že opravdu neexistuje orientace hran tak, aby vzniklý orientovaný graf D_2 měl nejvyšší odchozí stupeň nižší než 2. Platí proto

$$\Delta^+(G_2) = \Delta^+(D_2) = 2.$$



Obrázek 4: Orientované grafy D_1 (vlevo) a D_2 (vpravo)

■

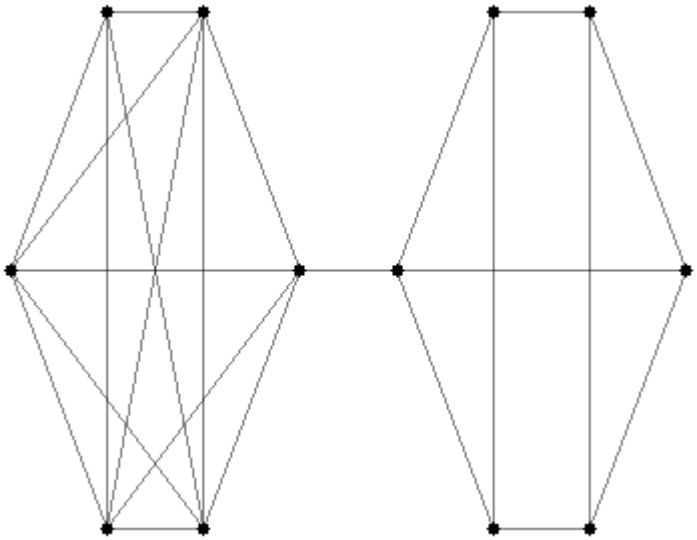
Dříve popsaný postup se sestrojením grafu G' a nalezením uzavřeného eulerovského tahu tedy u některých grafů nemusí najít optimální řešení.

Pokusme se určit kritéria, podle kterých budeme schopni rozhodnout, zda pro daný graf G bude platit $\Delta^+(G) = \left\lfloor \frac{d}{2} \right\rfloor$ nebo $\Delta^+(G) = \left\lceil \frac{d}{2} \right\rceil$. Už jsme ukázali, že pokud je v grafu G více vrcholů stupně d než vrcholů stupně $d-2$, řešení bude vždy $\Delta^+(G) = \left\lceil \frac{d}{2} \right\rceil$. Dále se proto budeme zabývat pouze grafy, v nichž tato podmínka není splněna.

Nejprve si uvědomme, že graf G nemusí být souvislý, což je právě případ grafu G_2 na Obrázku 4. Nesouvislý graf může obsahovat komponentu K, v níž jsou pouze vrcholy stupně d a $d-1$, případně pouze vrcholy stupně d, případně může být v komponentě K více vrcholů stupně d než vrcholů stupně $d-2$. Tuto komponentu K budeme řešit jako samostatný graf, čímž se dostáváme do už vyřešeného případu 3.2, případně 3.1 nebo na začátek aktuální podkapitoly a dostaneme opět řešení $\Delta^+(G) = \left\lceil \frac{d}{2} \right\rceil$.

Ani omezení na souvislé grafy nám ale pro jistotu dosažení $\Delta^+(G) = \left\lfloor \frac{d}{2} \right\rfloor$ nestačí. Předchozí odstavec můžeme ještě zobecnit. Nebudeme uvažovat pouze grafy, které obsahují komponentu K s vlastnostmi uvedenými v předchozím odstavci, ale všechny téměř pravidelné grafy, které obsahují libovolný indukovaný podgraf H s těmito vlastnostmi, tzn. $\delta(H) \geq d-1$ nebo je v podgrafu H více vrcholů stupně d než vrcholů stupně $d-2$. Je jasné, že platí $\Delta^+(H) = \left\lceil \frac{d}{2} \right\rceil$, a proto musí platit i $\Delta^+(G) \geq \left\lceil \frac{d}{2} \right\rceil$. Příkladem takového grafu je graf G se stupňovou posloupností $(5,5,5,5,5,4,4,3,3,3,3,3)$ na Obrázku 5, kde podgraf H získáme tak, že odebereme všech šest vrcholů, které jsou napravo od mostu. Dostaneme tak graf H se šesti vrcholy a stupňovou

posloupností $(5, 5, 5, 5, 4, 4)$, pro který určitě (viz Věta 5) platí $\Delta^+(H) = \left\lceil \frac{5}{2} \right\rceil = 3$. Není proto možné nalézt orientaci D, pro kterou by platilo $\Delta^+(D) = \left\lfloor \frac{5}{2} \right\rfloor = 2$. Proto platí $\Delta^+(G) = 3$.



Obrázek 5: Graf G s podgrafem H

Předpokládejme nyní, že pro všechny ostatní grafy G s vlastností $\Delta(G) - \delta(G) = 2$ platí $\Delta^+(G) = \left\lfloor \frac{d}{2} \right\rfloor$. Pokusme se orientovat hrany tak, abychom rovnost $\Delta^+(G) = \left\lfloor \frac{d}{2} \right\rfloor$ dokázali.

Opět upravíme graf G na eulerovský graf G', tentokrát ale jiným způsobem. Nebudeme do grafu žádné vrcholy ani hrany přidávat, naopak budeme hrany odstraňovat. Víme, že d je liché číslo, $d-2$ je tedy taky liché. Najdeme hranu mezi nějakým vrcholem stupně $d-2$ a nějakým vrcholem stupně d. Pokud v grafu G taková hrana není, pak najdeme cestu z nějakého vrcholu stupně $d-2$ do nějakého vrcholu stupně d tak, že na cestě mezi těmito vrcholy budou pouze vrcholy stupně $d-1$. V souvislém grafu G taková cesta určitě existuje, neboť v souvislém grafu existuje cesta mezi každými dvěma vrcholy. Pokud takovou cestu nenajdeme přímo z původně zvoleného vrcholu x a budeme nuceni v průběhu této cesty použít další vrchol y stupně $d-2$, pak namísto cesty z vrcholu x můžeme zvolit cestu z vrcholu y. Jakmile během cesty poprvé narazíme na vrchol stupně d, cestu v tomto vrcholu ukončíme. Proto v grafu musí být alespoň jeden vrchol stupně $d-2$, z kterého vede cesta do vrcholu stupně d pouze přes vrcholy stupně $d-1$.

Hrany z této cesty odstraníme. Odstraněním hran této cesty snížíme lichý stupeň d na sudý stupeň $d-1$ a lichý stupeň $d-2$ na sudý stupeň $d-3$. Z případných vrcholů sudého stupně $d-1$ na této cestě se po odstranění dvou incidentních hran stanou vrcholy opět sudého stupně $d-3$.

Namísto odstranění hran na uvedené cestě můžeme rovnou určit orientaci těchto hran. Při pozdějším hledání eulerovského tahu ale všechny orientované hrany budeme považovat za od-

straněné. Nemůžeme nic zkazit, když celou cestu orientujeme ve směru od vrcholu stupně $d-2$ k vrcholu stupně d. Pro lepší představu a snadnější zápis si v tuto chvíli za d dosaďme konkrétní číslo, například 7. Naším cílem je tedy najít takovou orientaci hran, abychom dokázali, že platí $\Delta^+(G) = \lfloor \frac{7}{2} \rfloor = 3$. Máme tedy nějaký téměř pravidelný graf G, pro který platí $\Delta(G) = 7$ a $\delta(G) = 5$. Orientováním cesty ve směru od vrcholu stupně 5 k vrcholu stupně 7 nic nezkazíme z toho důvodu, že vrchol stupně 5 má nyní jednu odchozí hranu a 4 neurčené (neurčené hrany budou později orientovány ve směru eulerovského tahu, 2 z nich tedy budou odchozí a zbylé dvě příchozí), vrchol stupně 6 má jednu příchozí hranu, jednu odchozí a 4 neurčené (opět nebude problém tyto neurčené hrany později orientovat pomocí eulerovského tahu tak, aby 2 byly příchozí a 2 odchozí), a nakonec vrchol stupně 7 bude mít jednu příchozí hranu a 6 neurčených. Po nalezení eulerovského tahu tedy všechny tyto vrcholy budou mít odchozí stupeň 3.

Nyní můžou nastat dvě různé situace.

- Nezbývá žádný vrchol stupně d. V tom případě orientujeme libovolně hrany mezi vrcholy stupně $d-2$ tak, abychom jejich stupně snížili na sudý stupeň $d-3$. Pokud v grafu taková hrana není, může to být i cesta, jejíž počáteční a koncový vrchol jsou stupně $d-2$ a zbylé vrcholy na cestě mají libovolný sudý stupeň). Případně můžeme nepoužité vrcholy stupně $d-2$ spárovat stejným způsobem jako v předchozích kapitolách. V tuto chvíli si to můžeme dovolit, neboť „problém“ by při spárování dělaly pouze vrcholy stupně d, které by pak po orientování podle uzavřeného eulerovského tahu měly více odchozích hran než požadujeme.

- Nějaké vrcholy stupně d zbývají a opět najdeme cestu z vrcholu stupně $d-2$ přes vrcholy sudých stupňů do vrcholu stupně d (tady už si nejsme jisti, jestli to vždy bude možné). Vrcholy stupně $d-2$ určitě zbývají také, jelikož jich na začátku bylo minimálně tolik jako vrcholů stupně d. Postup opakujeme, dokud nebude zbývat žádný vrchol stupně d.

Nakonec mají všechny vrcholy v upraveném grafu G' sudý stupeň a v grafu G' najdeme eulerovský tah. Pokud graf G' není souvislý, najdeme eulerovský tah v každé komponentě zvlášť. Po nalezení eulerovského tahu orientujeme hrany ve směru tahu a přidáme do grafu odstraněné hrany, které orientujeme výše popsaným způsobem po cestě směrem od vrcholu stupně $d-2$ k vrcholu stupně d.

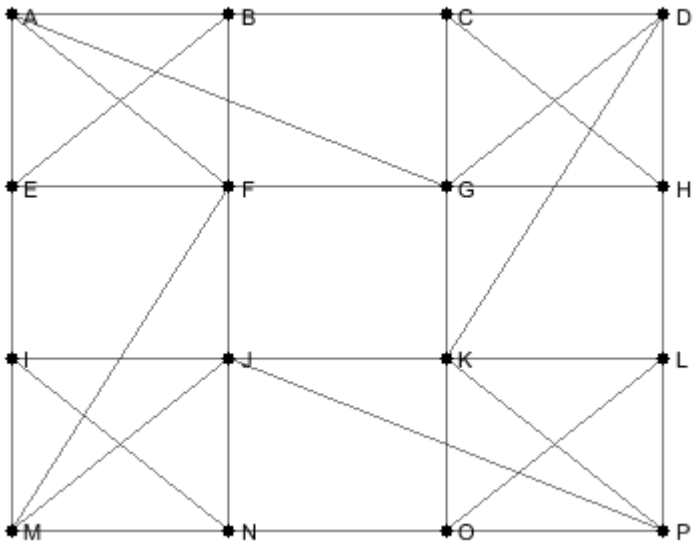
Pokud je v grafu G možné orientovat všechny hrany tímto způsobem, můžeme s jistotou říci $\Delta^+(G) = \left\lfloor \frac{d}{2} \right\rfloor$. V opačném případě (nejsme si jistí, je to pouze domněnka) bude platit $\Delta^+(G) = \left\lceil \frac{d}{2} \right\rceil$.

Tento problém prozatím ponecháme otevřený a vrátíme se k němu později na straně 39.

4 Hledání uzavřeného eulerovského tahu

V předchozí kapitole jsme zjistili, jakou hodnotu bude mít minimalizovaný nejvyšší odchozí stupeň $\Delta^+(G)$ v daném téměř pravidelném grafu G. Pro praktické využití ovšem potřebujeme nejen znát hodnotu $\Delta^+(G)$, ale také umět najít orientaci hran tak, abychom této hodnoty docílili.

Víme, že optimální orientaci hran umíme určit tak, že v grafu G, případně ve vhodně upraveném grafu G', nalezneme uzavřený eulerovský tah. Existují různé algoritmy pro nalezení uzavřeného eulerovského tahu. Nejznámějšími algoritmy jsou Fleuryho a Hierholzerův algoritmus. Oba tyto algoritmy z druhé poloviny 19. století si popíšeme a využijeme je k řešení konkrétních úloh.



Obrázek 6: Eulerovský graf

4.1 Fleuryho algoritmus

Francouzský matematik Fleury navrhl tento algoritmus v roce 1883 [4].

Algoritmus pro hledání uzavřeného eulerovského tahu začíná v libovolném vrcholu grafu G. V každém kroku vybereme libovolnou nezpracovanou hranu xy incidentní s aktuálním vrcholem x, která není mostem v grafu bez už zpracovaných hran a vrcholů, pokud taková hrana existuje (pokud neexistuje, vybereme most). Po této hraně se přesuneme do vrcholu y a hranu xy považujeme za zpracovanou. Z vrcholu y se stává aktuální vrchol a algoritmus takto pokračuje, dokud se nezpracují všechny hrany grafu G. Vrchol x považujeme za zpracovaný, jakmile nezbývají žádné nezpracované hrany incidentní s vrcholem x. Eulerovský tah je tvořen hranami v tom pořadí, v jakém byly zpracovány.

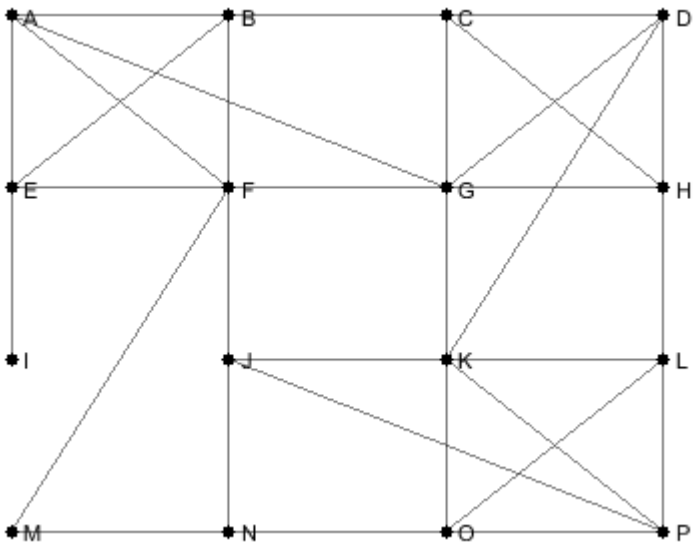
Postup algoritmu ukážeme také na příkladu.

Příklad 1

Pomocí Fleuryho algoritmu najděte uzavřený eulerovský tah v grafu na Obrázku 6.

Řešení: Snadno se můžeme přesvědčit, že všechny vrcholy grafu na Obrázku 6 jsou sudého stupně. Podle Věty 3 tedy v grafu existuje uzavřený eulerovský tah.

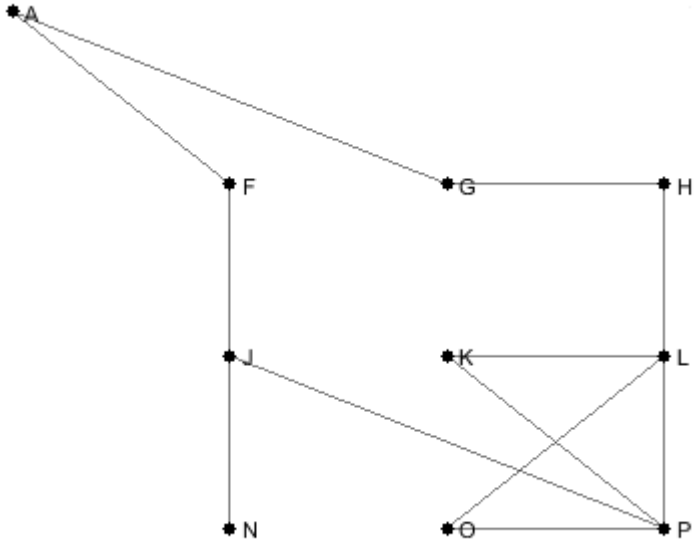
Můžeme zvolit libovolný vrchol, v němž tah zahájíme. Náhodně byl zvolen vrchol N. Zpracovávané hrany jsou taktéž voleny náhodně. Hranou IN se přesuneme do vrcholu I a hranu IN, která bude první hranou vznikajícího uzavřeného eulerovského tahu, považujeme za zpracovanou a z grafu ji odebereme. Dále vybíráme hrany IJ, JM, IM.



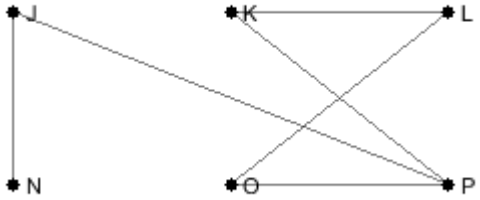
Obrázek 7: Graf po odebrání prvních čtyř hran

Všimněme si (viz Obrázek 7), že hrana EI je v aktuálním grafu mostem, neboť jejím odebráním se graf rozdělí na dvě komponenty. Jednou komponentou je samotný vrchol I, ostatní vrcholy a hrany tvoří druhou komponentu. Protože se ale jedná o jedinou nezpracovanou hranu incidentní s vrcholem I, v němž aktuálně jsme, nemůžeme zvolit jinou hranu, a proto použijeme hranu EI. Hranu EI a vrchol I z grafu odebereme. Tah dále pokračuje z vrcholu E hranami BE, BC, CG, GK, KO, NO, MN, FM, BF, AB, AE, EF, FG, DG, DH, CH, CD, DK, JK.

V tuto chvíli (viz Obrázek 8) zbývají tři hrany incidentní s vrcholem J. Nemůžeme použít hranu JN, neboť by vznikl nesouvislý graf, zatímco po použití hrany FJ nebo JP graf zůstane souvislý. Zvolíme tedy jednu z těchto dvou hran, například hranu FJ. Dále vynuceně volíme hrany AF, AG, GH, HL a dostáváme se do situace, kde si můžeme libovolně zvolit jednu ze tří zbývajících hran incidentních s vrcholem L. Zvolme hranu LP. Nyní máme opět tři hrany incidentní s aktuálním vrcholem (viz Obrázek 9). Jednu z nich ale zvolit nemůžeme. Kdybychom zvolili hranu JP, vznikl by nesouvislý graf, v němž by jednu komponentu tvořily vrcholy J, N a hrana JN.



Obrázek 8: Graf po odebrání dalších hran a vrcholů



Obrázek 9: Graf se 6 zbývajícími vrcholy i hranami

V předchozích nesouvislých případech během tohoto příkladu byla jedna z komponent vždy tvořena samotným vrcholem. Zde ale vidíme, že si nestačí dávat pozor na vznik triviálních komponent. To je důležité si uvědomit hlavně v případě, že budeme chtít Fleuryho algoritmus naprogramovat. Otestovat případný vznik triviální (tzn. jediným vrcholem tvořené) komponenty by bylo jednoduché. Ale zkontrolovat, jestli se souvislost grafu neporuší rozdělením grafu do libovolně velkých komponent, už je pro počítač podstatně složitější. Je nutné zjistit, jestli po odebrání hrany xy v grafu stále zbývá nějaká cesta mezi vrcholy x a y.

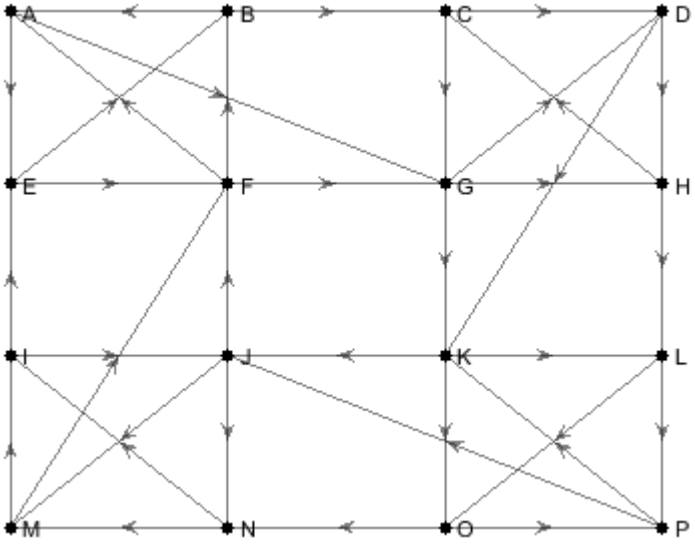
Pokračujme s tvorbou uzavřeného eulerovského tahu. Skončili jsme ve vrcholu P a víme, že nesmíme zvolit hranu JP. Zvolíme tedy třeba hranu KP. Dále už tah pokračuje až do konce jednoznačně danými hranami KL, LO, OP, JP, JN.

Nalezený uzavřený eulerovský tah nyní zapišme. Ze zápisu můžeme vynechat hrany, viz text pod Definicí 10. Nalezli jsme uzavřený eulerovský tah

$$t = (N, I, J, M, I, E, B, C, G, K, O, N, M, F, B, A, E, F,$$

$$G, D, H, C, D, K, J, F, A, G, H, L, P, K, L, O, P, J, N).$$

Na Obrázku 10 jsou hrany orientovány podle nalezeného eulerovského tahu.



Obrázek 10: Orientace hran - Fleuryho algoritmus

■

V nakresleném grafu obvykle jednoduše poznáme, jestli je daná hrana mostem, tedy jestli graf po odebrání této hrany zůstane souvislý. Algoritmus je proto poměrně jednoduchý pro manuální hledání eulerovského tahu, ale jak už bylo zmíněno, není příliš vhodný pro počítačové využití, kde se souvislost grafů zjišťuje složitěji. Jednodušším algoritmem z hlediska výpočetní složitosti je Hierholzerův algoritmus. Hierholzerův algoritmus má vzhledem k počtu hran lineární složitost, zatímco Fleuryho algoritmus má složitost polynomiální [5].

4.2 Hierholzerův algoritmus

Carl Hierholzer (1840 - 1871) byl německý matematik. Algoritmus vzniklý těsně před jeho smrtí byl publikován v roce 1873 [6].

Tento rekurzivní algoritmus pro hledání uzavřeného eulerovského tahu opět začíná v libovolném vrcholu. V každém kroku vybereme libovolnou nezpracovanou hranu incidentní s aktuálním vrcholem, přesuneme se pomocí této hrany do jejího druhého koncového vrcholu a hranu považujeme za zpracovanou. Jakmile se po několika krocích vrátíme do původního vrcholu (uzavřeme tah), nastávají dvě možnosti. Buď jsou všechny hrany zpracované a algoritmus končí, nebo nějaké hrany zbývají.

Pokud nějaké hrany zbývají, nalezený tah postupně procházíme a zastavíme se u prvního vrcholu x, který ještě není zpracovaný (je incidentní s nenulovým počtem nezpracovaných hran). Ve vrcholu x zahájíme další uzavřený tah. Po dokončení tahu opět zkontrolujeme, jestli ještě

zbývají nějaké nezpracované hrany. Pokud ano, najdeme v nejnověji nalezeném tahu první nezpracovaný vrchol a zahájíme v něm další tah. Jestliže všechny vrcholy nejnověji nalezeného tahu t_i jsou zpracované, prohledáme jeho předchůdce t_{i-1} a další tah zahájíme v prvním nezpracovaném vrcholu tahu t_i. Pokud ani v tahu t_{i-1} není nezpracovaný vrchol, pokračujeme opět do jeho předchůdce a případně do dalších předchůdců, dokud nenalezneme první nezpracovaný vrchol.

Nakonec všechny tahy spojíme dohromady. To uděláme tak, že každý tah t_i připojíme k jeho předchůdci t_{i-1} (tzn. k tahu, v němž jsme našli nezpracovaný vrchol x a zahájili v něm tah t_i). V tahu t_{i-1} najdeme první výskyt vrcholu x a v tomto místě tah t_{i-1} přerušíme a vložíme zde tah t_i. Po ukončení tahu t_i pokračujeme dalšími vrcholy tahu t_{i-1} následujícími po prvním výskytu vrcholu x.

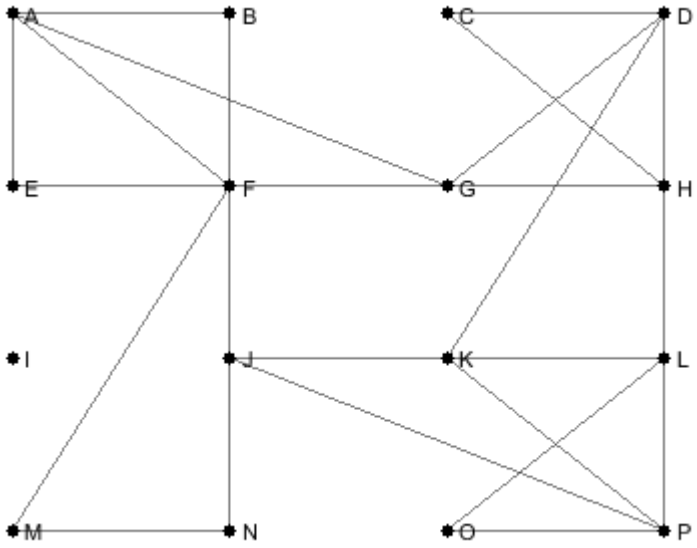
Poznámka 3 Uvedené označení t_{i-1} pro předchůdce bylo zvoleno pro snadnější popsání algoritmu. Není však úplně správné, neboť se může stát, že více tahů má stejného předchůdce. Bude-li t_i označovat tah nalezený jako i-tý v pořadí, předchůdce tahu t_i může mít někdy i nižší index než $i-1$. Například když nalezneme tah t_{i-1}, jehož předchůdcem je tah t_{i-2}, a v tahu t_{i-1} už poté nenajdeme žádný nezpracovaný vrchol, tak hledáme nezpracovaný vrchol v tahu t_{i-2}. Jestliže takový vrchol nalezneme a zahájíme v něm tah t_i, tak předchůdcem tahu t_i není tah t_{i-1}, ale tah t_{i-2}, jenž je zároveň předchůdcem i pro tah t_{i-1}.

I postup Hierholzerova algoritmu ukážeme na příkladu.

Příklad 2
Pomocí Hierholzerova algoritmu najděte uzavřený eulerovský tah v grafu na Obrázku 6.
Řešení: Snadno se můžeme přesvědčit, že graf na obrázku má všechny vrcholy sudého stupně. Podle Věty 3 tedy v grafu existuje uzavřený eulerovský tah.

Můžeme zvolit libovolný vrchol, v němž tah zahájíme. Vybíráme vrchol N stejně jako v případě Fleuryho algoritmu. I dále pokračujme stejně jako v případě Fleuryho algoritmu. S výběrem hran takto pokračujeme, dokud se nedostaneme zpět do vrcholu N. Vybrány jsou postupně hrany IN, IJ, JM, IM, EI, BE, BC, CG, GK, KO, NO. Dostali jsme se opět do počátečního vrcholu N. Máme uzavřený tah $t_1 = (N, I, J, M, I, E, B, C, G, K, O, N)$, ale ještě zůstávají nezpracované hrany, viz Obrázek 11.

Tento tah proto není eulerovský a algoritmus pokračuje. Nyní projdeme nalezený tah a zastavíme se u prvního vrcholu, který je incidentní s nějakými nezpracovanými hranami. To je hned vrchol N, začneme tedy další tah z vrcholu N a i nadále pokračujeme stejně jako u Fleuryho algoritmu. Vybíráme postupně hrany MN, FM, BF, AB, AE, EF, FG, DG, DH, CH, CD, DK, JK. Nyní se dostáváme do situace, kde se oba algoritmy odlišují. Můžeme zvolit hranu JN, kterou jsme v tuto chvíli během hledání uzavřeného eulerovského tahu pomocí Fleuryho algoritmu použít nemohli. Zkusme to tedy udělat. Dostáváme další uzavřený tah $t_2 = (N, M, F, B, A, E, F, G, D, H, C, D, K, J, N)$. Opět jsme se dostali do vrcholu N a opět



Obrázek 11: Graf po odebrání prvního uzavřeného tahu

zbývají nějaké nezpracované hrany. Žádná z nezpracovaných hran ale tentokrát není incidentní s vrcholem N, viz Obrázek 12.

Procházíme další vrcholy nejnověji nalezeného tahu, dokud se nedostaneme k prvnímu vrcholu, který je incidentní s nezpracovanými hranami. Všechny hrany incidentní s vrcholem M jsou také zpracovány, dostáváme se tedy do vrcholu F. V grafu jsou ještě dvě nezpracované hrany, s nimiž je vrchol F incidentní. Začneme tedy nový tah ve vrcholu F. Bereme postupně hrany FJ, JP, OP, LO, HL, GH, AG, AF a tah $t_3 = (F, J, P, O, L, H, G, A, F)$ je opět uzavřen. Stále zbývají nezpracované hrany, algoritmus proto pokračuje dále. První vrchol nejnovějšího tahu incidentní s ještě nezpracovanými hranami je vrchol P. Nový tah začneme v tomto vrcholu. Po hranách KP, KL a LP se dostáváme opět do vrcholu P a další tah $t_4 = (P, K, L, P)$ končí. Nyní už nezbývají žádné nezpracované hrany.

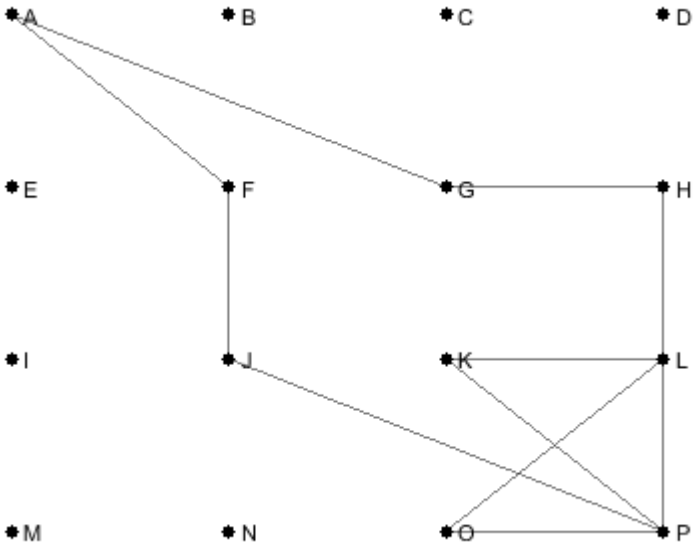
Jsou nalezeny čtyři tahy. Nejnověji nalezený tah t_4 připojíme k druhému nejnovějšímu tahu t_3. Tah t_4 začíná ve vrcholu P, k tahu t_3 jej proto připojíme ve chvíli, kdy se během tahu t_3 poprvé dostaneme do vrcholu P. Po připojení bude složený tah $t_{3,4}$ vypadat následovně: $t_{3,4} = (F, J, P, K, L, P, O, L, H, G, A, F)$. Tento tah připojíme ke druhému tahu a dostaneme tah $t_{2,3,4} = (N, M, F, J, P, K, L, P, O, L, H, G, A, F, B, A, E, F, G, D, H, C, D, K, J, N)$. Nakonec tah $t_{2,3,4}$ připojíme k prvnímu nalezenému uzavřenému tahu, čímž konečně vzniká uzavřený eulerovský tah

$$t = (N, M, F, J, P, K, L, P, O, L, H, G, A, F, B, A, E, F,$$

$$G, D, H, C, D, K, J, N, I, J, M, I, E, B, C, G, K, O, N).$$

Orientace hran podle nalezeného uzavřeného eulerovského tahu je zobrazena na Obrázku 13.

■



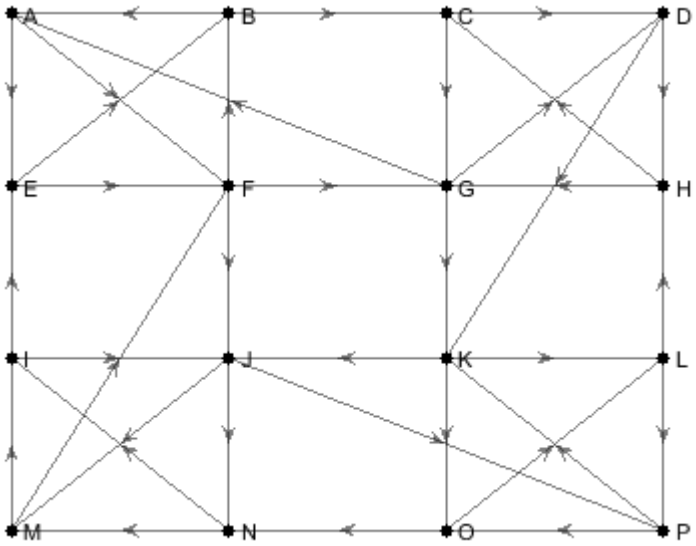
Obrázek 12: Graf po odebrání druhého uzavřeného tahu

Uvedeme zde také ukázku toho, jak lze Hierholzerův algoritmus naprogramovat v Matlabu.

```
function [circuit,D,G] = Hierholzer(G,first_vertex)

n = size(G,1);
D = zeros(n);
current_vertex = first_vertex;
circuit = first_vertex;
closed = false;

while ~closed
    for i=1:n
        if G(current_vertex,i) == 1
            G(current_vertex,i) = 0;
            G(i,current_vertex) = 0;
            D(current_vertex,i) = 1;
            circuit = [circuit i];
            current_vertex = i;
            break
        end
    end
    if current_vertex == first_vertex
        closed = true;
    end
```



Obrázek 13: Orientace hran - Hierholzerův algoritmus

```
end

n2 = length(circuit);

for i=1:n2-1
    if ~isequal(G(circuit(i),:),zeros(1,n))
        current_vertex = circuit(i);
        [circuit2,D2,G] = Hierholzer(G, current_vertex);
        D = D + D2;
        circuit = [circuit(1:i-1) circuit2 circuit(i+1:n2)];
    end
end

end
```

Výpis 1: Hierholzerův algoritmus v Matlabu

Vstupními parametry funkce `Hierholzer` jsou matice sousednosti jednoduchého eulerovského grafu G a první vrchol uzavřeného eulerovského tahu. Chceme-li, aby uzavřený eulerovský tah začínal ve vrcholu v_i, který je reprezentován i-tým řádkem v matici sousednosti grafu G, provedeme to příkazem `Hierholzer(G,i)`. Funkce vrací vektor, který reprezentuje posloupnost vrcholů v pořadí, v jakém jsou navštíveny během nalezeného uzavřeného eulerovského tahu (pro každé i od 1 do n, kde n je řád grafu G, je vrchol v_i v tomto vektoru vždy označen číslem i). Kromě tohoto vektoru může funkce vracet navíc ještě matici sousednosti oriento-

vaného grafu D, který vznikne orientováním hran grafu G ve směru nalezeného eulerovského tahu. To se nám bude hodit při hledání $\Delta^+(G)$. Chceme-li získat nejen nalezený eulerovský tah, ale i tuto matici, provedeme to příkazem `[circuit,D]=Hierholzer(G,i)`. Případně příkazem `[~,D]=Hierholzer(G,i)`, pokud nás zajímá pouze matice D.

V každém kroku algoritmu můžeme vybrat libovolnou hranu incidentní s aktuálním vrcholem. Algoritmus napsaný v Matlabu se mírně liší od algoritmu použitého v Příkladu 2 tím, že v Matlabu nevybíráme náhodnou hranu, ale první nalezenou. První nalezená hrana je vždy hrana vedoucí z aktuálního vrcholu v_i do vrcholu s nejnižším indexem i ze všech sousedních vrcholů vrcholu v_i.

5 Optimální orientování hran v téměř pravidelném grafu

Už víme, jak upravit téměř pravidelný graf G na eulerovský graf G' tak, aby platila rovnost $\Delta^+(G) = \Delta^+(G')$. Také víme, jak v eulerovském grafu najít uzavřený eulerovský tah. A nakonec víme, že orientováním hran podle uzavřeného eulerovského tahu v grafu G' dosáhneme optimální orientace - takové orientace, že pro vzniklý orientovaný graf D platí $\Delta^+(D) = \Delta^+(G)$. V této kapitole využijeme nabyté znalosti k řešení konkrétních úloh.

V minulé kapitole jsme si ukázali, jak je možné napsat Hierholzerův algoritmus v Matlabu. Pro řešení těchto úloh ale musíme zadaný téměř pravidelný graf G nejprve upravit na eulerovský graf postupem z Kapitoly 3. Prozatím vynecháme poslední probraný případ (grafy, v nichž platí $\Delta(G) - \delta(G) = 2$ a počet vrcholů stupně $\delta(G)$ je větší nebo roven počtu vrcholů stupně $\Delta(G)$), který budeme řešit jinak než ostatní případy, a podíváme se na možné řešení těch ostatních případů.

V Kapitole 3 jsme dokázali, že uvedené grafy je možné převést na eulerovské tak, že pro libovolnou dvojici vrcholů lichého stupně vytvoříme v grafu G nový vrchol a oba vrcholy lichého stupně s novým vrcholem spojíme hranou. Takto vytváříme nové vrcholy stupně 2, dokud v grafu zbývají nějaké vrcholy lichého stupně.

Před naprogramováním funkce pro takovou úpravu grafu G na graf G' si uvědomme, že do grafu často přidáváme zbytečně velké množství vrcholů. Přidáním nižšího počtu vrcholů snížíme paměťové nároky oproti původnímu návrhu, v němž jsme přidávali větší počet vrcholů stupně 2. Stačí přidat jediný vrchol. Všechny vrcholy lichého stupně s ním můžeme spojit hranou. Navíc tím snížíme na polovinu i počet přidaných hran. V grafu G' tak sice může být jeden vrchol vyššího stupně než všechny ostatní (tímto jedním vrcholem je právě přidaný vrchol) a graf D' vzniklý orientováním grafu G' tak nemusí splňovat rovnost $\Delta^+(D') = \Delta^+(G)$, ale to vůbec nevadí. Je totiž zřejmé, že pokud je $\Delta^+(D') > \Delta^+(G)$, tak nejvyšší odchozí stupeň má právě ten jediný přidaný vrchol. Ten ale odstraníme stejně jako jsme nově přidané vrcholy odstraňovali už dříve, čímž z grafu D' vytvoříme graf D. Pro graf D nyní platí $\Delta^+(D) = \Delta^+(G)$ a graf D tak představuje vhodnou orientaci hran grafu G.

```
function [D,DeltaplusD] = EdgeOrientation(G)

n = size(G,1);
DeltaplusD = 0;

for i=1:n
    degree = 0;
    for j=1:n
        if G(i,j)==1
            degree = degree + 1;
        end
```

```
    end
    if mod(degree,2)==1
        if size(G,1)==n
            G = [G;zeros(1,n)];
            G = [G zeros(n+1,1)];
        end
        G(i,n+1)=1;
        G(n+1,i)=1;
    end
end

[~,D]=Hierholzer(G,1);
D=D(1:n,1:n);
for i=1:n
    degree = 0;
    for j=1:n
        if D(i,j) == 1
            degree = degree + 1;
        end
    end
    if degree > DeltaplusD
        DeltaplusD = degree;
    end
end

end
```

Výpis 2: Orientování hran v Matlabu

Poznámka 4 Počet přidaných vrcholů by bylo možné snížit i na nulu. Stačí namísto přidávání vrcholů spárovat vrcholy lichého stupně a mezi dvěma spárovanými vrcholy vždy přidat hranu. Při takovém spárování se nám může stát, že přidáme hranu xy mezi vrcholy x a y, které byly sousední už v grafu G. V jednoduchém grafu nemůžou být dvě hrany xy, přidáním této hrany se z jednoduchého grafu G stává multigraf. V multigrafu může být mezi dvěma vrcholy x a y libovolný počet hran. Každá z těchto hran přispívá jedničkou do hodnoty $\deg(x)$ i $\deg(y)$. Věty o eulerovských grafech dokázané v Kapitole 2 pro jednoduchý graf můžeme ale stejným způsobem dokázat i pro multigrafy.

Pojďme nyní najít řešení pro grafy, u nichž předchozí algoritmus může namísto optimálního řešení nalézt řešení s o 1 vyšším nejvyšším odchozím stupněm, viz 3.3. V Podkapitole 3.3 byla bez

důkazu uvedena domněnka, jak by v takových grafech mohla být hodnota $\Delta^+(G)$ určena. Navrhli jsme mezi vrcholem x lichého stupně $\delta(G)$ a vrcholem y také lichého stupně $\Delta(G)$ nalézt cestu a všechny hrany na této cestě z grafu G odstranit. Takto tvoříme graf G' a pokračujeme hledáním dalších cest z vrcholů stupně $\delta(G)$ do vrcholů stupně $\Delta(G)$, dokud nesnížíme stupeň všech vrcholů stupně $\Delta(G)$. Pokud nezůstane v grafu G' žádný vrchol stupně $\Delta(G)$, zbývající vrcholy stupně $\delta(G)$ upravíme obvyklým způsobem pomocí hran do nově přidaného vrcholu. V každé komponentě vzniklého grafu G' nalezneme eulerovský tah a hrany orientujeme podle nalezeného eulerovského tahu, čímž vznikne graf D'. Následně nalezené cesty, které jsme odstranili při tvorbě grafu G', orientujeme ve směru z vrcholu x, který měl v grafu G stupeň $\delta(G)$, do vrcholu y, jenž byl původně stupně $\Delta(G)$. Přidáním těchto cest do grafu D' a odstraněním nových vrcholů vzniká graf D s optimální orientací hran.

Uvedený postup určitě funguje, pokud takové cesty nalezneme. Otázka je, jak bude vypadat řešení, pokud se nám tyto cesty nepodaří nalézt pro všechny vrcholy stupně $\Delta(G)$. Domněnka je taková, že pokud není možné uvedené cesty nalézt, optimální řešení úlohy bude o 1 vyšší než v případě, že v grafu takové cesty budou.

Tuto domněnku potvrzuje odkaz [3], kde je na straně 4 popsán algoritmus velmi podobný návrhu z naší domněnky. Uvedený algoritmus má najít optimální orientaci hran v libovolném grafu $G(V, E)$. V algoritmu se nejprve orientují náhodně všechny hrany grafu. Poté se najde vrchol s nejvyšším odchozím stupněm a hledá se orientovaná cesta (tzn. že hranu můžeme procházet jen v tom směru, kterým je orientovaná) do vrcholu, jehož odchozí stupeň je nižší alespoň o 2. Pokud taková orientovaná cesta v grafu je, orientaci všech hran této cesty otočíme a opět hledáme vrchol s nejvyšším odchozím stupněm a z něj orientovanou cestu s uvedenými vlastnostmi, kterou pak orientujeme opačným směrem. Takto pokračujeme, dokud se nedostaneme do situace, kdy nebude možné žádnou takovou orientovanou cestu najít. V tu chvíli se algoritmus ukončí a aktuální orientace hran je optimální.

Algoritmus naprogramujeme. Vytvořit počáteční orientaci hran není problém, najít vrchol s nejvyšším odchozím stupněm také ne. Jediný menší problém je nalézt algoritmicky nějakou cestu do vrcholu, jehož odchozí stupeň je alespoň o 2 nižší než nejvyšší odchozí stupeň. To můžeme vyřešit následovně. Prozkoumáme sousedy vybraného vrcholu. Pokud některý z nich splňuje uvedenou podmínku pro odchozí stupeň, cesta je nalezena. Pokud žádný z nich tuto podmínku nesplňuje, prozkoumáme sousedy těchto vrcholů, v případě nutnosti zase jejich sousedy (zkoumaného souseda vždy přidáme do cesty a vynecháváme jeho sousedy, které už jsme při cestě použili) atd. Nakonec mohou nastat dvě situace. Buď požadovaný vrchol nenalezneme a algoritmus končí, nebo jej nalezneme, změníme orientaci všech hran nalezené cesty a algoritmus se vrací k hledání vrcholu s nejvyšším odchozím stupněm v aktuálním grafu. Popsaný algoritmus se nazývá *prohledávání do šířky*, podobný algoritmus *prohledávání do hloubky* prozkoumává vrcholy v jiném pořadí. Začne v zadaném vrcholu, poté se přesune do jeho prvního souseda, pak do prvního souseda tohoto souseda atd. Do dalšího souseda zadaného vrcholu se dostane až po prozkoumání celé větve pod prvním sousedem.

Naprogramovaný algoritmus může vypadat následovně.

```
function [D,DeltaplusD] = AlgorithmReverse(G)

n = size(G,1);
D = zeros(n,n);
outdegrees = zeros(1,n);
changed = true;

for i=1:n
    for j=i+1:n
        if G(i,j) == 1
            orientation = rand(1);
            if orientation > 0.5
                D(i,j) = 1;
                outdegrees(i) = outdegrees(i) + 1;
            else
                D(j,i) = 1;
                outdegrees(j) = outdegrees(j) + 1;
            end
        end
    end
end

while changed == true
    DeltaplusD = 0;
    deltaplusD = n;
    for i=1:n
        if outdegrees(i) > DeltaplusD
            DeltaplusD = outdegrees(i);
            start = i;
        end
        if outdegrees(i) < deltaplusD
            deltaplusD = outdegrees(i);
        end
    end
    if DeltaplusD - deltaplusD < 2
        changed = false;
    else
        path = FindPath(D,start,outdegrees,n);
```

```
        plen = length(path);
        if outdegrees(path(plen)) > outdegrees(path(1)) - 2
            changed = false;
        else
            for i=1:plen-1
                D(path(i),path(i+1)) = 0;
                D(path(i+1),path(i)) = 1;
            end
            outdegrees(start) = outdegrees(start) - 1;
            outdegrees(path(plen)) = outdegrees(path(plen)) + 1;
        end
    end
end

end
```

Výpis 3: Algoritmus ze zdroje [3]

V cyklu `while` používáme rekurzivní funkci `FindPath` využívající prohledávání do hloubky, která vypadá následovně.

```
function [path] = FindPath(D,path,outdegrees,n)

dontuse = path;
stlen = length(path);
if outdegrees(path(stlen)) < outdegrees(path(1)) - 1
    return
end


for i=1:n
    if D(path(stlen),i) == 1
        if outdegrees(i) < outdegrees(path(1)) -1
            path = [path i];
            return
        end
    end
end

for i=1:n
    if D(path(stlen),i) == 1
```

```
        if ~ismember(i,dontuse)
            dontuse = [dontuse i];
            path = [path i];
            path = FindPath2(D,path,outdegrees,n);
            if outdegrees(path(end)) < outdegrees(path(1)) - 1
                return
            end
        end
    end
end

path(stlen) = [];

end
```

Výpis 4: Algoritmus pro nalezení cesty do vrcholu nízkého stupně

Algoritmus 3 můžeme zkusit upravit tak, že počáteční orientaci nebudeme volit náhodně, ale zvolíme ji pomocí způsobu, který využíváme pro ostatní téměř pravidelné grafy. Přidáme do grafu vrchol, spojíme jej hranami se všemi vrcholy lichého stupně, najdeme uzavřený eulerovský tah a přidaný vrchol zase odstraníme. Tím budeme po zvolení počáteční orientace podstatně blíže k optimálnímu řešení a k úpravě orientace budeme pravděpodobně potřebovat méně kroků než při náhodné orientaci. Porovnání algoritmů bude provedeno v následující kapitole.

```
function [D,DeltaplusD] = AlgorithmReverse2(G)

n = size(G,1);
D = EdgeOrientation(G);
outdegrees = zeros(1,n);
changed = true;

for i=1:n
    for j=1:n
        if D(i,j) == 1
            outdegrees(i) = outdegrees(i) + 1;
        end
    end
end

while changed == true
    DeltaplusD = 0;
```

```
    deltaplusD = n;
    for i=1:n
        if outdegrees(i) > DeltaplusD
            DeltaplusD = outdegrees(i);
            start = i;
        end
        if outdegrees(i) < deltaplusD
            deltaplusD = outdegrees(i);
        end
    end
    if DeltaplusD - deltaplusD < 2
        changed = false;
    else
        path = FindPath(D,start,outdegrees,n);
        plen = length(path);
        if outdegrees(path(plen)) > outdegrees(path(1)) - 2
            changed = false;
        else
            for i=1:plen-1
                D(path(i),path(i+1)) = 0;
                D(path(i+1),path(i)) = 1;
            end
            outdegrees(start) = outdegrees(start) - 1;
            outdegrees(path(plen)) = outdegrees(path(plen)) + 1;
        end
    end
end

end
```

Výpis 5: Upravený algoritmus ze zdroje [3]

6 Experimenty

Nyní, když máme vše naprogramované, můžeme v Matlabu otestovat rychlost naprogramovaných algoritmů a správnost obdržených výsledků. K tomu budeme potřebovat různé téměř pravidelné grafy, resp. matice sousednosti různých téměř pravidelných grafů. Aby nebylo nutné tvořit testovací grafy manuálně, což by pro grafy s velkým počtem vrcholů bylo velmi pracné, napíšeme v Matlabu funkci pro vytvoření matice sousednosti jednoduchého grafu s danou stupňovou posloupností.

Navržený algoritmus vychází z věty Havla-Hakimiho.

Věta 8 *Nechť $S = (s_1, s_2, \ldots, s_n)$ je nerostoucí posloupnost a nechť S' je posloupnost vzniklá z posloupnosti S tak, že odstraníme prvek s_1 a hodnoty dalších s_1 prvků (tedy prvků $s_2, s_3, \ldots, s_{s_1+1}$) snížíme o 1, pokud tyto prvky existují. Nakonec posloupnost přeuspořádáme na nerostoucí posloupnost. Pak S je stupňová posloupnost netriviálního grafu právě tehdy, když S' je stupňová posloupnost nějakého grafu.*

Důkaz věty je uveden například v [2].

Ukažme si na příkladu, jak můžeme z věty Havla-Hakimiho určit, zda je daná posloupnost grafová (tedy zda existuje graf G, pro který platí, že tato posloupnost je stupňovou posloupností grafu G).

Příklad 3
Rozhodněte, jestli je posloupnost $S = (5, 5, 5, 4, 4, 4, 3, 3, 3)$ grafovou posloupností.
Řešení: Vidíme, že $s_1 = 5$. Posloupnost S' dostaneme tak, že odstraníme s_1 a dalších 5 prvků snížíme o 1. Vznikne posloupnost $S' = (4, 4, 3, 3, 3, 3, 3, 3)$. Posloupnost už je seřazená nerostoucím způsobem, nemusíme ji tedy nijak upravovat. Posloupnost S je grafovou posloupností právě tehdy, když je grafovou posloupností i posloupnost S'. Stále ovšem nevíme, zda posloupnost S' je grafovou posloupností. Musíme tedy opět aplikovat stejný postup na posloupnost S' a budeme jej opakovat tak dlouho, dokud nebudeme schopni rozhodnout, zda je nová posloupnost grafová. Dostáváme postupně posloupnosti $(3, 3, 3, 3, 2, 2, 2)$, $(2, 2, 2, 2, 2, 2)$, $(2, 2, 2, 1, 1)$, $(1, 1, 1, 1)$, $(1, 1, 0)$, $(0, 0)$. Zde algoritmus ukončíme. Posloupnost $(0, 0)$ je stupňovou posloupností grafu, jenž se skládá ze dvou vrcholů, které nejsou spojeny hranou. Jedná se tedy o grafovou posloupnost a dle věty Havla-Hakimiho jsou grafovými posloupnostmi i všechny další posloupnosti vzniklé v průběhu algoritmu. I posloupnost S je tedy grafovou posloupností. ∎

Při řešení příkladu jsme mohli přestat tvořit další posloupnosti už dříve. Najít například graf se stupňovou posloupností $(2, 2, 2, 2, 2, 2)$ je jednoduché. Jedná se například o graf C_6, tedy cyklus se šesti vrcholy. Pokračovat až do odstranění nebo vynulování všech prvků má ale zejména při programování algoritmu smysl. Pokud se během algoritmu dostaneme k posloupnosti nul, zadaná posloupnost je vždy grafová. Pokud posloupnost není grafová, nedostaneme se k posloupnosti nul, ale k posloupnosti, která obsahuje -1, nebo k prázdné posloupnosti.

Následující algoritmus pro zadanou stupňovou posloupnost určí, jestli je grafová, a vypíše jednotlivé kroky jako řádky v matici $n \times n$, kde n je délka zadané stupňové posloupnosti. První prvek každé posloupnosti se při tvorbě matice neodstraní, ale pouze vynuluje. Na výsledné řešení to nemá vliv. Vliv by to mělo pouze u posloupností, kde $s_1 \geq n$. Takové posloupnosti nikdy nemůžou být grafové, neboť v grafu řádu n nemůže mít žádný vrchol více než $n-1$ sousedů, a tedy žádný vrchol nemůže mít ani stupeň větší než $n-1$.

```
function [steps,exist] = HavelHakimi(deg_seq)

exist = true;
n = length(deg_seq);
steps = zeros(n);
steps(1,:)=deg_seq;

if deg_seq(1)>=n
    exist = false;
else
    for i=1:n-1
        steps(i,:)=sort(steps(i,:),'descend');
        highest = steps(i,1);
        if highest > 0
            for j=2:highest+1
                steps(i+1,j-1)=steps(i,j)-1;
            end
            steps(i+1,highest+1:n-i)=steps(i,highest+2:n+1-i);
        else
            if steps(i,n)<0
                exist = false;
            end
            break
        end
    end
end

end
```

Výpis 6: Havel-Hakimi v Matlabu

Vraťme se k Příkladu 3. Zjistili jsme, že posloupnost $(5,5,5,4,4,4,3,3,3)$ je grafová. Když nyní budeme od konce procházet jednotlivé kroky uvedeného algoritmu, můžeme nějaký graf

s touto stupňovou posloupností vytvořit. Na příkladu si ukážeme, jak při tvorbě grafu postupovat.

Příklad 4

Vytvořte graf se stupňovou posloupností $(5, 5, 5, 4, 4, 4, 3, 3, 3)$.

Řešení: Poslední posloupností v Příkladu 3 byla posloupnost $(0, 0)$. Graf s touto stupňovou posloupností jednoduše vytvoříme - jedná se o dva vrcholy, které nejsou spojené hranou. Přesuneme se k předposlední posloupnosti, kterou byla posloupnost $(1, 1, 0)$. Takový graf vytvoříme z už vytvořeného grafu tak, že do něj přidáme jeden vrchol a ten spojíme hranou s jedním ze dvou původních vrcholů. Následující posloupnosti $(1, 1, 1, 1)$ docílíme tak, že do grafu přidáme další vrchol a spojíme jej hranou s jediným vrcholem, který je stupně 0. Posloupnost $(2, 2, 2, 1, 1)$ dostaneme tak, že do aktuálního grafu opět přidáme jeden vrchol a z něj hrany do libovolných dvou vrcholů ze zbývajících čtyř. Posloupnost $(2, 2, 2, 2, 2, 2)$ vznikne, když do grafu přidáme další vrchol a spojíme jej hranou s oběma vrcholy stupně 1. Přidáním dalšího vrcholu a tří hran z tohoto vrcholu do libovolných tří vrcholů dostaneme graf se stupňovou posloupností $(3, 3, 3, 3, 2, 2, 2)$. Další posloupností je posloupnost $(4, 4, 3, 3, 3, 3, 3, 3)$. Té docílíme tak, že do grafu přidáme další vrchol a ten spojíme hranou s jedním z vrcholů stupně 3 a všemi třemi vrcholy stupně 2. A nakonec stupňovou posloupnost $(5, 5, 5, 4, 4, 4, 3, 3, 3)$ dostaneme, když do grafu přidáme poslední vrchol a spojíme jej hranou s oběma vrcholy stupně 4 a třemi vrcholy stupně 3. ■

Obecně můžeme říci, že v každém kroku přidáme do grafu nový vrchol a hrany přidáváme vždy pouze tak, aby byly incidentní s tímto novým vrcholem. Jedná se vlastně o opačný postup než při zjišťování, zda graf s danou stupňovou posloupností existuje, kde jsme v každém kroku odebrali z (neznámého) grafu vrchol nejvyššího stupně a všechny hrany s ním incidentní.

Následující algoritmus v Matlabu nejprve zjistí jednotlivé kroky podobně jako v Příkladu 3 a následně tyto kroky prochází pozpátku, čímž postupně tvoří nějaký graf se zadanou posloupností. Ve chvíli, kdy je možné volit z více hran, se přidá náhodně vybraná hrana ze všech hran, které lze v dané situaci přidat. Algoritmus vrací graf se zadanou stupňovou posloupností, jenž můžeme využít k testování algoritmu pro nalezení optimální orientace hran.

```
function [G] = RandGraph(deg_seq)

[steps, exist] = HavelHakimi(deg_seq);
n = length(deg_seq);
G = zeros(n);
degrees=zeros(1,n);
remaining = 1:n;

if exist == true
    for i=n:-1:1
        changed = [];
```

```
        if steps(i,1) ~= 0
            chosindex = randsample((length(remaining)),1);
            chosen = remaining(chosindex);
            remaining(chosindex)=[];
            for j=2:n
                if steps(i,j)>steps(i+1,j-1)
                    changed = [changed steps(i+1,j-1)];
                end
            end
            l_changed = length(changed);
            for j=1:l_changed
                choosefrom = [];
                dontchoose = chosen;
                for k=1:n
                    if ~ismember(k,dontchoose) && degrees(k)==changed(j)
                        choosefrom = [choosefrom k];
                    end
                end
                if length(choosefrom)>1
                    edgeto = randsample(choosefrom,1);
                else
                    edgeto = choosefrom;
                end
                G(chosen,edgeto)=1;
                G(edgeto,chosen)=1;
                degrees(edgeto)=degrees(edgeto)+1;
                degrees(chosen)=degrees(chosen)+1;
                dontchoose = [dontchoose edgeto];
                l_rem = length(remaining);
                for k = 1:l_rem
                    if remaining(k) == edgeto && degrees(remaining(k))==1
                        remaining(k)=[];
                        break
                    end
                end
            end
        end
    end
end
```

```
end
```

Výpis 7: Tvorba náhodného grafu v Matlabu

Pravděpodobě je možné vymyslet i kvalitnější algoritmy, jak vytvořit náhodný téměř pravidelný graf. Uvedený algoritmus neumožňuje vytvořit jakýkoli graf se zadanou stupňovou posloupností. Pro potřeby této práce ale tento algoritmus postačí. Při řešení reálných úloh nebude potřeba vytvářet náhodné grafy, tudíž není nutné uvedený algoritmus zdokonalovat.

Matice sousednosti vytvořené tímto algoritmem nyní můžeme využít pro otestování napsaných algoritmů. Každý algoritmus bude pro každou matici sousednosti opakován desetkrát, přičemž do tabulky zapíšeme nejlepší čas, nejhorší čas a průměr všech časů. Všechny časy jsou uvedeny v sekundách.

Začneme grafem, který má 50 vrcholů stupně 11, 150 vrcholů stupně 10 a 70 vrcholů stupně 9. Tabulka pro tento graf vypadá následovně.

	Nejnižší čas	**Nejvyšší čas**	**Průměrný čas**
EdgeOrientation	0.0426	0.0525	0.048
AlgorithmReverse	0.0291	0.0982	0.0532
AlgorithmReverse2	0.0513	0.0709	0.0603

Tabulka 1: Graf s 50 vrcholy stupně 11, 150 vrcholy stupně 10 a 70 vrcholy stupně 9

Algoritmus `EdgeOrientation` byl sice v průměru nejrychlejší, ale dokázal najít pouze řešení o 1 horší než optimální ($\Delta^+(G) = 5$, nalezeno řešení s nejvyšším odchozím stupněm 6). `AlgorithmReverse` měl poměrně velký rozptyl výsledků, což je vzhledem k náhodnému počátečnímu orientování hran logické. Tento algoritmus pokaždé nalezl optimální řešení stejně jako `AlgorithmReverse2`, který měl menší rozptyl časů, ale v průměru byl pomalejší. Všechny algoritmy byly ovšem velmi rychlé, rozdíly v časech potřebných k výpočtu jsou zde tedy bezvýznamné.

Dalším otestovaným grafem bude graf s 250 vrcholy stupně 25, 500 vrcholy stupně 24 a 250 vrcholy stupně 23.

	Nejnižší čas	**Nejvyšší čas**	**Průměrný čas**
EdgeOrientation	1.0436	1.0755	1.0577
AlgorithmReverse	0.3862	0.8622	0.6223
AlgorithmReverse2	1.3146	1.3461	1.3289

Tabulka 2: Graf s 250 vrcholy stupně 25, 500 vrcholy stupně 24 a 250 vrcholy stupně 23

I v tomto grafu bylo optimální řešení s nejvyšším odchozím stupněm 12 nalezeno pouze pomocí algoritmů `AlgorithmReverse` a `AlgorithmReverse2`, zatímco algoritmus `EdgeOrientation` nalezl orientace s nejvyšším odchozím stupněm 13. `AlgorithmReverse` byl navíc v průměru téměř dvakrát rychlejší než `EdgeOrientation`.

Dále algoritmus vyzkoušíme na grafu s 500 vrcholy stupně 55 a 500 vrcholy stupně 56. Tentokrát už se tedy jedná o graf, pro který nalezne optimální orientaci ($\Delta^+(G) = 28$) i algoritmus `EdgeOrientation`. Počet hran tedy bude více než dvojnásobný oproti předchozímu příkladu, zatímco počet vrcholů bude stejný. Algoritmus `AlgorithmReverse2` nemá smysl používat, neboť už počáteční orientace daná nalezeným uzavřeným eulerovským tahem bude optimální. Využijeme tedy pouze algoritmy `EdgeOrientation` a `AlgorithmReverse`.

	Nejnižší čas	**Nejvyšší čas**	**Průměrný čas**
EdgeOrientation	2.1107	2.1497	2.1273
AlgorithmReverse	0.7854	1.7297	1.0682

Tabulka 3: Graf s 500 vrcholy stupně 56 a 500 vrcholy stupně 55

Dalším grafem bude graf s 1000 vrcholy stupně 13 a 1000 vrcholy stupně 12. Algoritmus `AlgorithmReverse2` opět nemá smysl používat. Algoritmus `AlgorithmReverse` nedokázal při žádném z 10 pokusů nalézt řešení, vždy byl dosažen maximální povolený počet rekurzí. V následující tabulce bude proto pouze algoritmus `EdgeOrientation`.

	Nejnižší čas	**Nejvyšší čas**	**Průměrný čas**
EdgeOrientation	2.7827	2.8124	2.7975

Tabulka 4: Graf s 1000 vrcholy stupně 13 a 1000 vrcholy stupně 12

V grafech s velkým počtem vrcholů a malým počtem hran bude pro algoritmus `AlgorithmReverse` obtížnější nalézt cestu do požadovaného vrcholu. Může se tak stát, že je během hledání cesty naší rekurzivní funkcí `FindPath` aloritmus ukončen Matlabem dříve, než dojde k výsledku, jak jsme viděli v případě tohoto grafu. Pro podobné grafy tedy bude rozumné využít náš algoritmus `EdgeOrientation`, v němž by podobný problém nastat neměl, nebo vylepšit prohledávání grafu při hledání cesty. Při rozdělení objektu na velký počet podoblastí budou vznikat zpravidla právě takové duální grafy a stupně vrcholů budou často ještě nižší, viz Obrázek 2, kde je čtverec rozdělen do 16 menších čtverců a každý menší čtverec do dvou trojúhelníků. Všechny vrcholy duálního grafu jsou stupně 1, 2 nebo 3. Když si představíme, jak by vypadal duální graf pro rozdělení do více menších čtverců (a opět každý čtverec do dvou trojúhelníků), zjistíme, že vrcholů bude mnohem více, ale stupně vrcholů budou pořád 1, 2 a 3.

Víme, že pro některé grafy (pro grafy splňující současně podmínky $\Delta(G) - \delta(G) = 2$, liché $\Delta(G)$ a počet vrcholů stupně $\delta(G)$ větší nebo roven počtu vrcholů stupně $\Delta(G)$) nenalezne náš algoritmus `EdgeOrientation` optimální řešení, ale pouze řešení o 1 horší. To nás ovšem u grafů, kde je objekt rozdělen do velmi vysokého počtu podoblastí, nemusí trápit, neboť takové grafy budou mít prakticky vždy vrcholy vyšších stupňů „vevnitř“ a takových vrcholů bude při rozdělení do velkého počtu podoblastí obvykle mnohem více než „vnějších“ vrcholů nižšího stupně. Optimální orientace v praxi používaných duálních grafů tedy bude prakticky vždy řešitelná pomocí algoritmu `EdgeOrientation`.

7 Závěr

V diplomové práci se podařilo vyřešit zadané téma. Nakonec jsme ale postupovali úplně jiným způsobem, než který byl naznačen v zadání práce. Téměř pravidelné grafy jsme rozdělili do několika typů a s pomocí vlastního algoritmu v Matlabu umíme velmi rychle určit optimální orientaci hran pro všechny typy kromě jednoho. Poslední typ téměř pravidelných grafů jsou grafy s vlastností $\Delta(G) - \delta(G) = 2$ a počet vrcholů stupně $\delta(G)$ je větší nebo roven počtu vrcholů stupně $\Delta(G)$, přičemž $\delta(G)$ je liché. Náš algoritmus umí najít optimální řešení pouze pro některé grafy tohoto typu, zatímco pro jiné najde řešení o 1 horší než optimální. S tímto typem grafů se ovšem v praxi (o plánovaném praktickém využití jsme se zmiňovali už v úvodu) pravděpodobně moc často nesetkáme, neboť vrcholy nejnižšího stupně se v duálních grafech budou při rozumných rozděleních objektu na podoblasti vyskytovat zpravidla pouze na krajích. Vnitřní podoblasti, kterých obvykle bude více, budou mít téměř vždy více sousedních podoblastí, tudíž vrcholy, kterými budou vnitřní oblasti reprezentovány, budou vyššího stupně. Pokud se přece jen s takovým grafem v praxi setkáme, můžeme se rozhodnout, zda se spokojíme s případnou o 1 horší orientací hran než je optimální orientace, nebo zkusíme využít jiný algoritmus, který je znám už delší dobu a v této práci je také popsán a naprogramován.

Kódy v Matlabu určitě nejsou napsány dokonale a je možné je různými způsoby vylepšit ať už z hlediska využití paměti (množství využité paměti lze snížit velmi výrazně, pokud využijeme například řídké matice) nebo z hlediska toho, že některé výpočty se pravděpodobně zbytečně opakují (tady už ale možnosti vylepšení asi nejsou tak velké jako v případě paměti). Cílem bylo především napsat kódy, které budou funkční a dostatečně rychlé, což se víceméně podařilo. Zdokonalování kódů je možné v případě, že se algoritmy budou využívat v praxi.

Práce navazuje na další práci katedry aplikované matematiky v oblasti minimalizace nejvyššího odchozího stupně, kterou se v současné době zabývá tým ve složení Matěj Krbeček, Petr Kovář, Michal Kravčenko a Adam Silber. Jejich práce je zaměřena částečně na rovinné grafy a částečně i na obecnou minimalizaci v libovolném jednoduchém grafu.

Literatura

[1] CHROBAK, Marek a David EPPSTEIN. Planar Orientations with Low Out-Degree and Compaction of Adjacency Matrices [online]. [cit. 2017-04-26]. Dostupné z: https://www.ics.uci.edu/~eppstein/pubs/ChrEpp-TCS-91.pdf

[2] KOVÁŘ, Petr. Teorie grafů [online]. [cit. 2017-03-20]. Dostupné z: http://homel.vsb.cz/~kov16/files/skriptum_teorie_grafu_rozsirene.pdf

[3] ASAHIRO, Yuichi, Eiji MIYANO, Hirotaka ONO a Kouhei ZENMYO. Graph Orientation Algorithms to Minimize the Maximum Outdegree [online]. [cit. 2017-04-26]. DOI: 10.1.1.60.3168. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=3268E68C584274B5085C154E71A5D82E?doi=10.1.1.60.3168&rep=rep1&type=pdf

[4] OH, Sewoong. Overview - euleriancycle.pdf. In: *University of Illinois Urbana-Champaign* [online]. [cit. 2017-03-24]. Dostupné z: http://swoh.web.engr.illinois.edu/courses/ie512/handout/euleriancycle.pdf

[5] Eulerian path. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-03-24]. Dostupné z: https://en.wikipedia.org/wiki/Eulerian_path

[6] Carl Hielholzer. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-03-24]. Dostupné z: https://en.wikipedia.org/wiki/Carl_Hierholzer